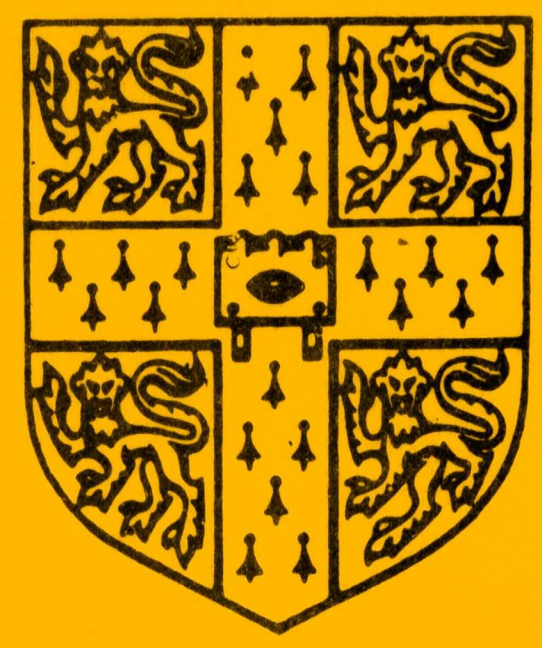
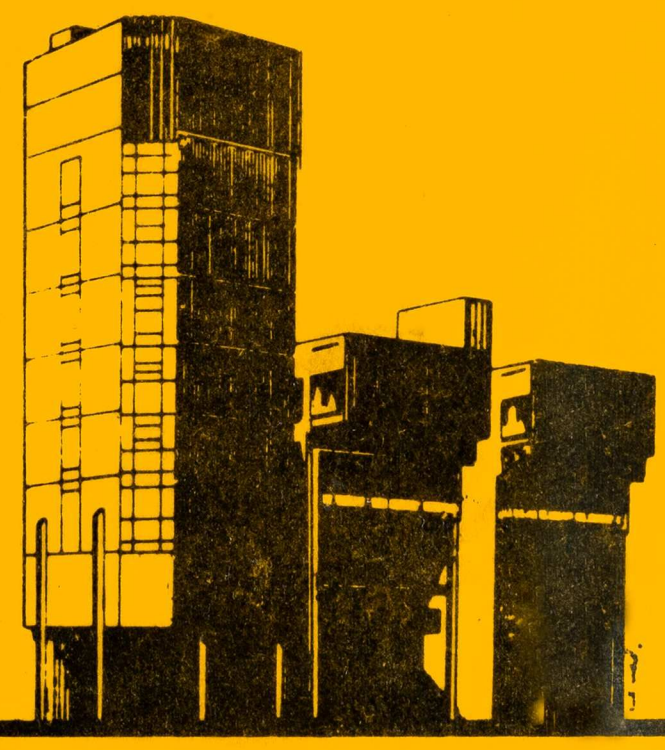
UNIVERSITY of CAMBRIDGE
COMPUTING SERVICE



CAMBRIDGE 370/165

USERS' REFERENCE MANUAL



University of Cambridge
Computer Laboratory
Corn Exchange Street
Cambridge CB2 3QG
England

UNIVERSITY OF CAMBRIDGE COMPUTING SERVICE

CAMBRIDGE 370/165

USERS' REFERENCE MANUAL

3rd Edition

with Amendments 3

Editors:-

P. Stewart and R. J. Stibbs

March 1978

CONTENTS

PART I

1. INTRODUCTION

2. ACCESS TO THE SYSTEM

2.1 System structure

2.2 Phoenix consoles

2.2.1 Communication lines and consoles
2.2.2 Console resource management
2.2.3 Line connection
2.2.4 LOGON
2.2.5 Character and record handling
2.2.6 Console session
2.2.7 FINISH
2.2.8 System failures
2.2.9 Special console features

2.3 Jobs

2.3.1 Job types
2.3.2 Phoenix jobs
2.3.3 JCL jobs
2.3.4 Hasp terminators

2.4 Job descriptions

2.4.1 Job statement

2.5 Job description statements

2.6 System output

2.6.1 Phoenix and JCL jobs
2.6.2 Phoenix Monitor

2.7 Resource allocation and accounting

2.7.1 Resource control
2.7.2 Resource units

2.8 Job limits and restricted facilities

2.8.1 Job limits
2.8.2 Restricted facilities

3. DATA MANAGEMENT

3.1 Magnetic disc storage

3.1.1 Temporary disc datasets
3.1.2 The filing system
3.1.3 Permanent disc space control
3.1.4 Safety and security of datasets
3.1.5 Private discs
3.1.6 BIGDISC

3.2 Magnetic tapes

3.2.1 Hardware
3.2.2 Administration of magnetic tapes
3.2.3 Use of magnetic tape
3.2.4 Names and labels
3.2.5 Jobs
3.2.6 Tape parameters on Data Definition statements
3.2.7 Tape marks
3.2.8 Scratch tapes
3.2.9 Transfer of magnetic tapes
3.2.10 Magnetic tape write-permit rings

3.3 DCB parameters, size parameters and access methods

3.3.1 Access methods and dataset types
3.3.2 Blocking factors
3.3.3 Simplified DCB statements for JCL
3.3.4 Phoenix size qualifiers

3.4 Paper tape

3.4.1 Paper tape input
3.4.2 Input widths and codes
3.4.3 Error messages
3.4.4 Warning sequences
3.4.5 Options
3.4.6 Job submission
3.4.7 Paper tape output
3.4.8 Output codes and widths
3.4.9 Jobs

3.5 Peripheral devices

3.5.1 Plotter
3.5.2 Printers

4. PHOENIX LANGUAGE

4.1 Syntax of Phoenix basic commands

4.1.1 Control commands
4.1.2 Program execution commands

4.2 Syntax of command-environments

4.2.1 Store requirement
4.2.2 Time requirement
4.2.3 Parameter data
4.2.4 Default settings

4.3 Data definitions

4.3.1 Syntactic elements
4.3.2 Special names
4.3.3 Qualifiers

4.4 Examples of complete basic Phoenix commands

4.5 Semantics of data definitions

4.5.1 Devices
4.5.2 Catalogued disc datasets
4.5.3 Temporary disc datasets
4.5.4 Workspace disc datasets
4.5.5 Ddname datasets
4.5.6 System documents
4.5.7 Members of datasets
4.5.8 Instream (here) datasets
4.5.9 Concatenations
4.5.10 Ddname lists

4.6 Physical characteristics of datasets

4.6.1 Size
4.6.2 Format characteristics
4.6.3 Status
4.6.4 Organisation
4.6.5 Other characteristics

4.7 Macro commands

4.8 Command sequences

4.8.1 C-commands
4.8.2 Procedures
4.8.3 Levels and nesting

4.9 Parameters

4.9.1 Parameter names
4.9.2 Parameter values
4.9.3 Parameter substitution
4.9.4 Local and global settings
4.9.5 Special parameters

4.10 Control features

4.10.1 Abends
4.10.2 Flags and switches
4.10.3 Control commands
4.10.4 Labels and jumps
4.10.5 Conditional commands
4.10.6 Other control commands
4.10.7 Flow of control after error

4.11 Miscellaneous

4.11.1 Interrupting commands
4.11.2 SET commands
4.11.3 SET CHAR command
4.11.4 Initial state
4.11.5 Provision of extra macro libraries

APPENDICES

Appendix A Character Codes Tables

Appendix B Standard Escape Combinations

Appendix C Printer Chain

Appendix D Phoenix Online Interface

Appendix E Phoenix Macro Commands

Appendix F Scheduling

Appendix G Restricted Utilities

Appendix H Carriage Control Characters

Appendix I Using the GT40

PART II

SPECIFICATIONS OF COMMANDS AND PROCEDURES

INDEX

1. INTRODUCTION

The purpose of this reference manual is to present, in conjunction with relevant IBM and other Cambridge publications, information on the use of the Cambridge 370/165 installation. It is a reference and not a tutorial manual and therefore presupposes some knowledge of the system. Users without this knowledge are urged to attend one of the courses arranged by the Computing Service. In any case, new users of the Cambridge installation, whether or not they have experience of other IBM 360 or 370 computers, should first consult the publication 'Basic Use of the Cambridge 370/165' which describes the basic concepts of both the IBM system and the extensions developed in Cambridge. Indeed, 'Basic Use' will be sufficient for many simple applications of the computer.

The Cambridge 370/165 installation is based on an IBM System 370 Model 165 processor with 3 Mbytes of main memory and a large capacity file store (disc) shared between the operating system and users' files (or datasets). Magnetic tape units and various slow peripheral devices are provided, enabling information to be transferred into and out of the system by means of magnetic tape, punched cards, paper tape, printers, and a graph plotter. In addition, a Digital Equipment Corporation PDP11 configuration connected to the 370/165 provides communications facilities enabling a large number of devices to be connected to the installation, including Remote Job Entry (RJE) terminals and multiple access (Phoenix) consoles. Details of the configuration and of other computing equipment available are given in the publication 'An Introduction to the University of Cambridge Computing Service'. Details of the Graphics dedicated system are given in the 'PDP 11/45 - Vector General Display Graphics System Manual'.

The 370/165 is operated under the IBM 360 Operating System (OS MVT Release 21) together with the 'Houston Asynchronous Spooling and Priority' program (commonly known as HASP). A number of extensions and improvements have been developed by the Computing Service, notably the Phoenix multiple-access system with its command language, and the mechanisms for resource allocation and job scheduling. However, an essential degree of compatibility between the Cambridge system and other IBM 360 and 370 installations is maintained to minimise the work involved in transferring programs and data. Inevitably there are always some differences between installations which affect users, and there are some features of the standard system described in IBM publications that are either different or unavailable at a particular installation. The relevant information for the Cambridge 370/165 is given in this manual.

Access to the facilities of the system is described in Chapter 2. Work may be performed either directly (online) at a Phoenix console or by submission of jobs; since the facilities available online are restricted, a combination of these methods is commonly used, it being possible both to submit jobs and to receive output at a Phoenix console. The resource allocation and control scheme for scheduling the order of execution of jobs by means of project shares is also described.

Chapter 3 is concerned with details of data management including the storage of information on disc and tape, access to this information by programs, and the handling of input and output via punched cards, paper tape, etc.

The Phoenix command language, which is used both online at Phoenix consoles and as an alternative to Job Control Language (JCL) in jobs, is defined in Chapter 4. Appendices at the end of Part I include additional reference material.

Part II (Specifications) contains detailed reference material on the use of the various commands, procedures and languages available on the 370/165.

The attention of users is drawn to supplementary information contained in other Cambridge publications. A series of Example Sheets is available which illustrates the use of individual service and language procedures. Full details of all available Cambridge publications and of relevant IBM publications are given in the 'Catalogue of Documentation'. All publications may be obtained from the Computing Service Bookshop.

Amendments to this manual will be issued periodically and sent to registered users. However, for up-to-date information on any subject covered in this manual Phoenix users should type HELP 'subject' NEW for information on a particular subject. JCL users should list the file INFO.NEW.

2. ACCESS TO THE SYSTEM

2.1 SYSTEM STRUCTURE

The essential characteristic of the system software of the Cambridge 370/165 is that it is a multi-access system. IBM's OS MVT and HASP systems have been modified and extended to enable a large number of jobs to be processed with widely varying degrees of size and urgency; at the same time, the system aims to provide users with convenient mechanisms by which to get their work done with the aid of Phoenix consoles and Remote Job Entry (RJE) terminals. An important policy in developing the installation has been to support online consoles to the extent that most users, whether or not their work is of the traditional 'batch processing' type, would naturally use consoles as a means of access to the system.

The operating system differs from standard OS/HASP in four basic ways:

(a) A number of restrictions are imposed on the use of certain features of OS; these are mainly to ensure efficiency of the installation as a whole.

(b) The system has been extended in a number of ways to provide additional facilities, utility programs, etc. The user, of course, must realise that these extensions are unlikely to be found at other installations.

(c) A number of internal alterations have been made to improve the performance of the system in the Cambridge environment; however, these do not directly affect users.

(d) The catalogued procedures have been altered. Part II of this manual (Specifications) gives details of the various procedures.

The Phoenix console system is a major Cambridge development that has been based on IBM's Time Sharing Option (TSO) which is a component of OS. The users' command language has been redesigned and therefore Phoenix appears quite different to the user. The Phoenix command language is also available in jobs as an alternative to JCL. The type of work that can be performed at a Phoenix console is generally limited, and the normal user will find that he has available a range of standard facilities mainly for file handling and job submission. However, all facilities including the ability to execute users' own programs are available in jobs. Any job may be submitted from a Phoenix console, and many jobs can be processed while the user waits at a console. General instructions for operating a Phoenix console are given in Section 2.2.

Jobs can be submitted to the operating system from a number of different sources. Irrespective of the source the output of a job can be sent to different destinations. The possible job sources are as follows:

(a) Phoenix consoles. The Phoenix language provides a number of commands, which, when typed at a Phoenix console, enable a user to submit a job to the job queue via the internal reader as described in (e) below.

(b) Reception reader. A card reader located in Computing Service Reception and operated by users.

(c) Central readers. These include card and paper tape readers located in the 370/165 computer room and operated by the Computing Service staff.

(d) Remote readers. These are associated with RJE (Remote Job Entry) terminals; usually a card reader is provided, but sometimes a paper tape reader is also available. More complicated arrangements may exist at some remote sites, but each RJE station generally appears to the 370/165 as one or more readers.

(e) Internal readers. These are the part of the system that enables a program running in the 370/165 to submit a new job. Thus one user's job may create and submit another.

The possible output destinations are:

(a) Central printers and punches. There are two line printers in Computing Service Reception and one in the 370/165 computer room together with a card punch, two paper tape punches and a Calcomp 1136 graph plotter.

(b) Remote stations. Every RJE station has a line printer and some of them have card and paper tape punches.

(c) Disc files. The complete output of a job may be sent to a disc file from Phoenix jobs by using the SPOOL command (see Part II). Output destined for the plotter may be diverted to a disc file, and then examined on a graphics console using the VIEW command (see Part II).

(d) Most of the output may be suppressed by using the ROUTE DUMMY job description statement (see Section 2.5).

Two types of job are handled by the operating system; Phoenix jobs and JCL jobs. Details of both types follow in Section 2.4.

A common feature of every job irrespective of its type is that it begins with a job description (JD) which is not compatible with the job cards accepted by other OS/HASP installations. A JD is used to supply certain information concerning the job as a whole; its format is defined in Section 2.4. It replaces both the IBM job card and also the various local cards found at some other IBM installations, for example /*SETUP, /*PRIORITY, /*PASSWORD etc. In particular the JD indicates the type of job (Phoenix or JCL), the resources needed by the job (amount of store, cpu time, tapes, printer output etc.), and routing

information for the output. All jobs submitted to the operating system are held for execution in the job queue which is organised by a Cambridge-developed scheduler in HASP. The scheduler is concerned with the (often conflicting) requirements of maintaining a proper balance in the use of resources by different user projects, attempting to meet users' needs concerning the turnround of their jobs, and ensuring the overall performance of the installation. Details are given in Appendix F.

2.2 PHOENIX CONSOLES

2.2.1 COMMUNICATION LINES AND CONSOLES

Phoenix consoles are of various types having different features and operating at different speeds; they are connected to the 370/165 via the PDP11 communications system employing a variety of equipment and systems software. In every case a console is connected to the communications system by means of a line having certain characteristics. The communication line provides either a permanent connection between the console and the system (in the case of private or Post Office leased connections) or a temporary one (in the case of dialled connections via the Post Office Switched Telephone Network). The lines and consoles operate asynchronously with transmission speeds between 110 baud and 1200 baud. Dialled connections are at present supported at 110, 134.5, 150, 200 and 300 baud. Consoles may be of various types and the system is capable of supporting the following:

(a) Standard 64-character ASCII devices
Typically these are Teletype devices operating at 110 baud, but other devices including simple visual display units (VDU's) with the same characteristics will usually work.

(b) 96-character ASCII devices
As above but providing a larger range of characters including two cases of letters. Both types of ASCII device can be supported on lines operating at speeds up to 1200 baud.

(c) IBM 2741 Consoles
These operate at 134.5 baud.

(d) Graphics consoles, which may be used for the VIEW command (see Part II), such as the Tektronix models 4006, 4010 and 4012 display terminals or any programmable terminal running a suitable emulator, such as that available for the DEC GT40.

Consoles operate either in full-duplex or half-duplex mode. In full-duplex mode all printing on the console is controlled by the communications system and as each character is sent from the console to the PDP11 the same character is normally echoed back from the PDP11 to the console for printing or display. Half-duplex consoles print or display each character as it is transmitted, and therefore it is essential that the PDP11 system does not echo back. Permanent lines may be full or half-duplex; dialled lines operate in full-duplex until the console user indicates otherwise (see Appendix D).

Users who intend to obtain a console and possibly a permanent line to the system must consult the Computing Service in advance to check that the device to be used can be supported and to arrange for the necessary line characteristics to be provided.

2.2.2 CONSOLE RESOURCE MANAGEMENT

The process of an individual user accessing the system via a Phoenix console is known as a console session. The user first finds a console that is not in use, proceeds to log on, conducts his business with the system, and finally logs off before leaving the console. The logon procedure (described below) is necessary both to establish the identity of the user and to ensure that it is possible for the system to give the service required by the user at that time. The user must log off when he has finished to enable the system resources required to support a console session to be freed for other users. Consoles which have no associated CPU activity receive a warning after 5 minutes and, if left idle for a further 5 minutes, are automatically logged off. At the present time the system is relatively simple: a fixed number of console sessions can be in progress simultaneously (the number varies according to operational and systems considerations), users are not charged for the resources used by the session itself, and in general only one kind of service is provided. Future developments will provide more comprehensive resource control facilities, and it is also possible that more than one kind of service will be available for a console session in the future. Meanwhile, users are generally provided with a standard mode of access. An alternative mode is in fact available but since it is currently used only for system maintenance purposes it is not covered in this manual.

2.2.3 LINE CONNECTION

A console that is not attached to the communications system by a permanent communication line must, on each occasion it is used, be connected to the system by use of a suitable Post Office modem or acoustic coupler and telephone, and by dialling the installation through the Public Switched Telephone Network. The telephone number is:-

Cambridge (0223) 66401 (5 lines)

(If an ASCII terminal is other than of the standard type, the user should make use of the @D mechanism (see Appendix D) to set the appropriate characteristics once the logon process has started or after it has been completed.) The system automatically answers the call and,

provided the communications system is running, the ringing tone will change to a data tone (a continuous high-pitched note) within two or three seconds. The system will then wait for 10 seconds to allow the user to connect an acoustic coupler or to switch to 'data' as appropriate; after this the system will continuously monitor the 'carrier detect' line from the user's modem and if at any time the carrier is found to be absent, the system will hang up the call and the user will be logged off. This happens if the user disconnects his modem, but it is good practice always to logoff (see below) before doing this.

2.2.4 LOGON

The logon procedure is started by a user typing 'return' on a console that is connected to the communications system and not already in the logged on state. (If the console is not connected to the system it 'chatters'. If it is already logged on for another user, the FINISH command must first be typed - see 2.2.7 below.) The system will normally respond with

LOGON

after which the user should type his user identifier (see Section 2.4) terminated by 'return' or 'esc'. If the userid has not been typed after 30 seconds ***ABANDONED is typed and the LOGON sequence is terminated. Errors may be corrected as described below using @C, @L etc. Certain alternative or additional responses may be given by the system as follows:

(a) No response at all - either the console is not connected or the communications system is not in operation.

(b) PHOENIX NOT AVAILABLE - although the communications system is working, the 370/165 is not currently supporting console sessions or is about to close down.

(c) Operator alternative - this may give helpful information as to why Phoenix is not available.

Whenever a new version of this message is put into the communications system, it is automatically typed on all switched on consoles.

(d) ONLINE SYSTEM FULL - the maximum permitted number of simultaneous console sessions has been reached; the user should try again a little later.

The message "PHOENIX STARTING" is typed on all switched on consoles when the Phoenix online system is started.

After the userid has been typed, the system may reply

(a) USERID IN USE - this indicates that a session is in progress at some other console for the user identifier given; only one session per user identifier is allowed at any one time. If the use of the identifier is illegal, Reception should be notified immediately (Extension 600).

(b) ENTER VALID USERID - the userid typed is not a Phoenix user. This may be due to a typing error.

If the system is able to allow the console session to begin, it will execute the user's initialisation sequence (see MKINIT in Part II) and might continue with a 'Message of the day', which is usually a brief report on such facts as the service schedule, changes to the system, etc. This is followed by the standard response

READY

indicating that the console session is in command state and that the user may proceed with his work. If the user wishes to abandon (and possibly begin afresh) the logon procedure, he may do so by pressing 'break' before terminating his response to LOGON; otherwise this can be done once the session has started by use of the FINISH command. The user may interrupt and abandon typing of the message of the day, by similar use of 'break'.

2.2.5 CHARACTER AND RECORD HANDLING

The communications system accepts characters from a console in the code of the console, reflects them back to the console (in the case of full-duplex working) and translates the code to EBCDIC. The translated characters are stored within the communications system until a complete record (i.e. line) has been received; the record is then transmitted to Phoenix in the 370/165. Similarly output for a console is received from Phoenix one whole record at a time, stored in the communications system and then translated and transmitted to the console character by character. The code translation process depends on the character set for the console, and precisely what happens in each case is specified in the tables of Appendix A.

As a general rule, the user terminates each record with 'return' and the system responds with 'carriage return, line feed'. On input from a console a general escape mechanism operates both to extend the character set of the console and to provide special control and signalling characters. The character @ is always treated as the escape character, and instead of being input in the normal way causes one or more (but usually one) characters immediately following to be treated specially. Two particular uses of this are @C, which deletes the preceding character, and @L, which deletes the whole current line. Note that the character @ may be input by using the combination @@. Full specifications of the escape combinations are given in Appendix B.

The translation rules from EBCDIC are given in Appendix A.

As soon as the communications system has finished transmitting an input record to Phoenix, further input from the console is accepted; there may thus be a short period between one input record and the next when input is not accepted.

When output for the terminal arrives from the 370/165, the PDP-11 behaves as follows:-

(a) If the user has not typed any valid characters since the end of the last input or output line, the PDP-11 stops accepting input, and proceeds immediately to the output.

(b) If the user has typed valid characters, the PDP-11 allows 60 seconds for the completion of the input record. If this deadline is met, the input is sent off to the 370/165 while the input is taking place. Otherwise '!!carriage return, linefeed' is output, and the incomplete input record is abandoned.

When output is taking place, the system will not accept input from the console, but if required the user can take action in one of the following ways:

(a) Typing 'break' or '@I' causes ! to be typed, the remainder of the current output record to be lost, and a 'break' signal to be sent to Phoenix.

(b) Typing 'space' causes the communication system to pause at the end of the current output record for 60 seconds. This is useful for visual display units. Typing '@L' cancels the pause immediately. Alternatively, a line of input may be typed which is sent to Phoenix in the 370/165 while subsequent output takes place.

2.2.6 CONSOLE SESSION

Once the standard READY response has been received, the console session is in command state, and the user may type any single command available according to the mode of the session. Full details of the Phoenix command language will be found in Chapter 4 and specifications of the individual commands in Part II (Specifications). When a command is complete the system, with only one exception, returns the console to command state indicating this by typing READY. Note that sequences of commands can be obeyed together through a special command called the C (or COMMAND) command. (See Section 4.8.1 and Part II).

Two facilities are provided that enable messages to be sent to a user while he is logged on. Firstly, there is a mechanism whereby the operations or systems staff may send the same message to all currently logged-on consoles. This may be used to give immediate warnings of system problems, scheduled or unscheduled imminent shut-down of the service etc. The second facility enables one user to send a message to another on an individual basis. For further details see the MESSAGE, NOTIFY and SEND commands in Part II. When a new 'Message of the day' is put into the system a line

***NEW MESSAGE OF THE DAY

is usually typed at all logged on consoles. Users should type MESSAGE DAY to receive the message.

2.2.7 FINISH

The FINISH command is used to terminate a console session; this logs off the user and frees the console and its communication line for subsequent use by another person. Details of the FINISH command are given in Sections 4.10.3 and 4.10.6 and in Part II (Specifications).

2.2.8 SYSTEM FAILURES

When a system failure occurs, the user will first find that typing carriage return has no effect. Within a few minutes one of two things should happen:-

(a) ***ERROR
is typed at the console. This means that there has been a failure in the communications system, and the user may immediately continue with his console session. The line he was typing will normally have been lost.

(b) ***ABANDONED
is typed at the console, and the response to carriage return will be a line such as
PHOENIX NOT AVAILABLE
The console session has been cancelled because of a catastrophic failure and all temporary datasets are lost. Usually the system will be restarted after an interval and
PHOENIX STARTING
will be typed. The user must then recommence the LOGON process.

2.2.9 SPECIAL CONSOLE FEATURES

(a) Line width
The default line width of a console is set to 72 characters for ASCII devices (the carriage width of a teletype) although some vdu's on fixed lines have a line width of 80 characters and 120 characters for IBM 2741 terminals. Any output records longer than this are spilled onto the next line. Users who require lines of a different length may make use of the Phoenix command SET TERMINAL WIDTH. (See Part II (Specifications).)

(b) Non-ASCII features
It is important to note that some devices (notably IBM 2741 and those using full 96-character ASCII) differ from the standard in their characteristics. The differences are small but can be important and users of these consoles should consult the detailed specifications given in Appendix D. Graphical features are described under the VIEW command (see Part II).

(c) Console paper tape input
Some consoles are fitted with paper tape readers and punches which can usually be switched manually to transmit to, or receive from, Phoenix. Paper tape punching presents no particular problems to the system or the user. However, because both Phoenix in the 370/165 and the communications system are designed primarily for typing at normal human speeds, a number of problems arise with the use of paper tape input, and these are more likely to occur when higher line speeds are used. In particular there is no means by which the system can cause the console reader to stop transmitting characters, and since it is possible, though unlikely, for system congestion to occur, the mechanism cannot be guaranteed to be 100% perfect. Thus, although the INPUT command (see Part II) may be used to read in pre-prepared documents, users must find out in practice the extent to which this method is sufficiently reliable, particularly with respect to regular and sustained use. Note also that all normal character conventions apply to input from paper tape, including the escape mechanism. Illegal characters and combinations are faulted in the usual way, and messages may break in without warning unless SET TERMINAL NOMESSAGE is set. The @DH combination (see Appendix D) may be used on a full-duplex device to suppress normal character reflection, but 'return' always gives rise to the 'carriage return, line feed' response.

2.3 JOBS

2.3.1 JOB TYPES

Two different types of job are accepted by the Cambridge 370/165.

(a) Phoenix jobs use the Phoenix command language. Users may find the Phoenix language more convenient to use than JCL, particularly since it is also used by the console system. Some facilities are not yet available in the Phoenix command language, although many of these will be added in due course. Details of how to write Phoenix commands and procedures to use various facilities are given in Part II, and the basic syntax of the language is defined in Chapter 4.

(b) JCL jobs use the OS Job Control Language (JCL) which is described in the JCL Reference Manual (GC28-6704). Most facilities are available to a JCL job. Installation dependent features of JCL are described in Section 2.3.3 below. Details of the various procedures are given in Part II.

There is also a Phoenix Monitor system which runs small jobs more efficiently. A Phoenix job will normally be run through the Phoenix Monitor automatically if it is small enough (see Section 2.8.1).

Although the automatic system will work for most users, there may be a few jobs that satisfy all the conditions for the Phoenix Monitor, but do not run satisfactorily under it (for example, because the time allowed for transfers of data to and from disc may be exceeded). The job description statements DISCIO and NOPMON are provided to allow for manual overriding of the system (see Section 2.5).

All jobs begin with a standard Job Description as defined in Section 2.4. The essential characteristics of each type of job are given in the following sections, and the standard form of output from a job is described in Section 2.6. Details of the resource limits imposed on jobs are described in Section 2.8.1. The availability of, and method of access to, particular service and language facilities are given in Part II.

2.3.2 PHOENIX JOBS

Phoenix jobs consist of a Job Description that has been terminated by PHOENIX, PHX or * followed by one or more lines of commands written in the Phoenix Command language. This is described in Chapter 4 with examples in Part II. If JCL statements appear in a Phoenix job they must immediately follow the JD terminator PHOENIX.

Command reflections and messages from a Phoenix job are sent to a dataset called COMMOUT0. The default destination for this dataset is * (normally the printer) but output may be redirected by use of the SPOOL command (see Part II). For a complete list of the ddnames used in a Phoenix job see Section 4.3.2.

2.3.3 JCL JOBS

A JCL job consists of a Job Description terminated by the JD parameter JCL, followed by one or more lines of JCL as defined in the IBM publication 'Job Control Language Reference Manual' GC28-6704.

The following gives particular information concerning the Cambridge installation:-

Group names may be specified for the UNIT parameter in a dd statement as follows:

Unit	Group name and synonyms
3330 disc	DISC (DISK)
7-track tape	TAPE7
9-track tape	TAPE9 (TAPE)
Internal reader	INTRDR

The internal reader enables one job to submit another to the system. This is normally done by using the SUBMIT utility (see Part II) to transmit to the internal reader a dataset containing a complete card image of a job. Users who wish to write to the internal reader directly are advised to begin the job with the HASP terminator /*JOB... and to end it with the HASP terminator /*EOF, to ensure that the job is not confused with any other.

Possible values or the SYSOUT parameter in a dd statement are:

Unit		Value	
Printer	(	A	(see below)
	(	C	
Card punch		B	
Plotter		P	(see Section 3.5.1)
Wide plotter		Q	(see Section 2.8.2)
Special plotter		R	(see Section 2.8.2)
Paper tape punch (5-track)		5	(see Section 3.4)
Paper tape punch (7-track)		7	
Paper tape punch (8-track)		8	

SYSOUT=A provides normal printer output, and in particular page throws are not altered in any way. SYSOUT=C provides a more economical use of paper by converting all skip carriage control characters into three space carriage control characters. This is particularly useful since many compilers and other programs are profligate in their use of paper. The Cambridge catalogued procedures employ this output class in default in appropriate circumstances.

Sections 3.3.2 and 3.3.3 give recommendations for BLKSIZE and for simplified DCB parameters.

2.3. HASP TERMINATORS

The following terminators, which are interpreted by HASP, may appear in jobs input from any source (card readers, paper tape readers or the internal reader). They must start in column 1 of a record.

- /*DEL — deletes the current job, if any, on the reader. This is useful to clear a card reader after, for example, a card jam.
- /*EOF — terminates the current job, if any, on the reader. This terminator should always be used when several jobs are stacked together in the reader.
- /*JOB... — terminates the current job, if any, on the reader and is then treated as a normal job statement. Users are advised to use this, rather than JOB, in a job statement for internal readers.

A terminator is also available which enables the characters /*, at the beginning of a record, to be read in. This facility is available via the consultancy service.

2.4 JOB DESCRIPTIONS

The Job Description (JD) that must be provided at the head of every job submitted to the Cambridge 370/165 is not compatible with the JOB cards required by other OS/HASP installations. When transferring a program from or to another installation, this part at least will have to be altered.

A JD consists of a sequence of statements each terminated by a comma or end of record. The layout is free format, in that additional spaces may be inserted at the beginning and end of a statement and wherever a space is specified as optional or mandatory. Sequence numbers (in character positions 73-80) are permitted if the last statement on each line is terminated by a comma. Characters beyond character position 80 are ignored.

The first statement in the JD is a mandatory JOB statement; the remainder are all optional, and enable various parameters of the job to be set; in many cases, omission of these parameters gives rise to default settings. In each case a parameter is specified by its name, followed by = and/or one or more spaces, followed by a value. Note that lower-case letters are permitted as substitutes for upper-case letters. A JD as a whole is terminated in a manner according to the type of the job (PHOENIX or JCL). All job description statements are defined in the next section.

Thus, the format of a job description is:

```
        JOB statement
        Job description statements
        Job description terminator (PHOENIX or JCL)

e.g.    JOB SPQR 2345,COMP 6 SECS
        TURNROUND BY 1630 OR REJECT
        PHOENIX
```

2.4.1 JOB STATEMENT

The JOB statement begins with the word JOB, followed by a space and the user identifier, then a space and the project number, then a space and the job title.

If a default project has been set by using the PROJECT command (see Part II) then the project number in the JOB statement may be replaced by *.

The user identifier is a four character string, the first character of which is alphabetic, and is used both to construct a unique name for the job and to ensure that any printed output finds its way back to the user. The project number is an integer and is the authorisation code for the job. Both the user identifier and the project number are issued

by the Computing Service. When a job is submitted to the 370/165 the userid and project number are checked to ensure that the use of the project is legal. If the use is illegal the job is rejected. The job title (which may be absent) is a string of characters chosen by the user which does not include comma or ampersand. It is copied to the output of the job, but truncated to a maximum of 18 characters.

Examples of JOB statements:

```
JOB SPQR 100 TEST JOB 26
JOB ABC1 123 ANYTHING
JOB PQRS 176
```

2.5 JOB DESCRIPTION STATEMENTS

Job description statements describe the overall system resources needed for the job, the job type, route information, security and scheduling information. Each statement may be on a separate line or statements may be on the same line separated by a comma.

The statements may be divided into several classes:-

Job size:-	COMP, LIMSTORE
Scheduling:-	TURNROUND, PRIORITY, CHAIN
Security:-	PASSWORD
Output requirements:-	COPIES, PLOTTER, PUNCH, PRINTER, PAGE, ROUTE, POST, NOTIFY, MSGLEVEL, MSGCLASS
Additional requirements:-	TAPE9, TAPE7, DISC, BIGDISC, DECKS, COND
Special requirements:-	DISCIO or NOPMON
Job description terminator:-	PHOENIX, JCL

Details of these statements follow in alphabetical order.

BIGDISC

This parameter is used to give access to large amounts (more than 2000 tracks) of temporary disc space in a job. See BIGDISC in Chapter 3.1.6.

Example:

```
BIGDISC
```

N.B. This statement may only be used after special authorisation.

CHAIN

This statement when used in conjunction with the PRIORITY and TURNROUND statements allows a user to control the order in which a set of jobs is run. Jobs which contain the CHAIN statement and which have the same user identifier will be run one at a time in the order in which they appear in the job queue. No chained job will be initiated (taken from the job queue in order to be executed) if another job, chained or not, for the user identifier is currently executing. Note that the order of running is not necessarily the same order in which the jobs were submitted, as a change of priority or usage may cause a later job to leapfrog other already queued jobs. To ensure that jobs run in a specific order, users are advised to use TURNROUND AT in conjunction with CHAIN.

Example:
```
JOB RJS4 444
CHAIN
PHX
C ...
```

COMP

This statement controls the amount of cpu time allowed for the complete job. The value is an integer followed by any number of spaces and the units SEC, SECS, SECONDS, MIN, MINS, MINUTES. The default unit is SECONDS.

The default value is 5 SECS and the maximum value is 20 MINS. If a job requests more than 5 seconds it will not be run through the Phoenix Monitor.

Examples:
```
COMP 1 MIN

COMP 15 MINS
```

TIME is a synonym for COMP

COND

This is similar to the corresponding JCL keyword COND, as described in the JCL Reference Manual. The COND specified in the job description applies to all steps of the job. Note that Phoenix jobs are always one job step from the JCL point of view.

Example:
```
JOB SPQR 9999,COND=(4,LE)
```

COPIES

This statement should be used with great discretion. It enables multiple copies of printer output to be produced. The value is the number of copies. The default is 1. The printer limit applies to the sum of the lines output and not individual copies. Note that if the printer limit is inadequate several incomplete copies of the document will be produced.

Example: COPIES 2

DECKS

The DECKS (or DECK) statement may be used to tell the system that the job requires fewer 7-track or 9-track decks than the number of 7-track or 9-track tapes it uses. Both the TAPE and DECKS statements should be used when the number of tapes is greater than the number of decks. The DECKS statement is unnecessary if the two are equal.

Examples: DECK 9 2

DECKS 7 1

N.B. The charge for a tape job depends on the number of decks that it requires.

DISC

This parameter is used to specify a private disc pack. It is followed by the name of the pack. DISK is a synonym.

Example: DISC PRIDIS

N.B. This statement may only be used after special authorisation.

DISCIO

This statement is used to tell the system that the job is NOT to be run through the Phoenix Monitor, because it needs more time for transfers to and from disc than the Phoenix Monitor allows. No value is required.

Example: DISCIO

JCL

This statement if present, must be the last statement of the JD. It defines the job as a JCL job.

Example:
```
JOB RJS9 999
JCL
// EXEC , ...
    etc
```

LIMSTORE

This statement gives the maximum amount of store needed by any job step. The integer value is in bytes and may be followed by K which represents units of 1024 bytes. This statement must be present if the region requested by any job step is greater than 120K The default value of LIMSTORE is 120K and the maximum value is 400K A job which requires more than 130K will not be run through the Phoenix Monitor.

Example LIMSTORE 150K

MSGCLASS MSGLEVEL

These are the same as the corresponding JCL parameters described in the JCL Reference Manual GC28-6704

Example: MSGLEVEL=(1,1)

NOPMON

This statement should be included whenever the job should not be processed by the Phoenix Monitor. NOPMON has the same effect as DISCIO

Example NOPMON

NOTIFY

This statement is used to send a message from a job to a Phoenix user when the job finishes It may be optionally followed by a user identifier which may be different from that in the JOB statement. If the user is not logged in at the time the message is sent, it is lost NOTIFY is also a Phoenix command, see Part II

Note that the Phoenix command and the JCL catalogued procedure SEND are more useful for notification of job progress than NOTIFY. (If the job is scheduled for running through the Phoenix Monitor, NOTIFY is ignored).

Examples:	NOTIFY	sends a message to the user identifier on the JOB statement
	NOTIFY ABC1	sends a message to ABC1

PAGE

The PAGE statement is used to control the number of lines per page. A zero value stops HASP interpreting page throw carriage control characters. The default is 60. Some utilities interrogate the value of the PAGE statement.

Examples: PAGE 0

PAGE 40

PASSWORD

The PASSWORD statement may be used to protect users against illegal use of their resource allocations. A password consists of a string of letters and digits of length less than or equal to eight characters. To set and quote a password use the command

PASSWORD password1:password2

either parameter may be null. For a new user the password is set to null If a user has a password which is not null, a job submitted under his userid will be rejected with the message

BAD JOB - INCORRECT PASSWORD

unless it includes the correct password. A password statement is not required if the password is null. If a user forgets a password he may apply to Reception to have it reset. Note that the password is linked to the user identifier, not the project number.

Examples:	PASSWORD :ABC102	sets a previously unset password.
	PASSWORD ABC102	quotes the password, but does not change it.
	PASSWORD ABC102:ABC103	quotes the password then sets a new one.
	PASSWORD ABC103:	sets the password to null.

PHOENIX

This statement, if present, must be the last statement of the JD. It defines the job as a Phoenix job. (See Section 2.3.2). PHX and * are synonyms for PHOENIX. Note that if the job is small enough (see 2.8.1 below) the job will be run through the Phoenix Monitor, unless the statement DISCIO or NOPMON is included in the Job Description.

Example:
```
JOB ABCD 7777,PHOENIX
EXAMINE
        etc.
```

PLOTTER

The PLOTTER statement is used to define the amount of plotter output that a job will produce. The unit is RECORDS. The default value is zero. The maximum value is 10K. The sum of the PUNCH and PLOTTER records declared may not exceed 10K.

Example:
```
PLOTTER 100 RECORDS
```

POST

This statement enables output to be directed to an output tank that belongs to a different user or pseudo-user from the one identified by the userid on the job statement. The value is the destination userid. Output may also be posted to pseudo-users. e.g. WCAV sends output produced at the central site to West Cambridge via the Computing Service Van; output sent to MAIL will be posted by Reception staff to the user after prior arrangement

Example:
```
POST RJS8
```

The ROUTE statement (see below) enables output to be redirected for printing and punching at another location.

PRINTER

This statement gives the limit of the number of lines of printer output which the job produces, in total. The integer value may be followed by K representing a unit of 1024 and/or the qualifier LINES. The default is 500 and the maximum is 80K.

Examples:
```
PRINTER 2K

PRINTER 300 LINES
```

PRIORITY

The PRIORITY statement specifies an integer value between 1 and 255 inclusive. PRIORITY specifies an upper limit to the priority at which the job will be run and charged. The default value depends on the size of the job. Users are advised to read Appendix F before using this statement.

Example: PRIORITY 5

PUNCH

This statement gives the limit of the number of card or paper tape punch records to be produced by the job. The integer value may be followed by K, representing a unit of 1024, and/or the qualifier CARDS. The default is zero and the maximum is 10K. The sum of the PUNCH and PLOTTER records declared may not exceed 10K.

Example: PUNCH 200 CARDS

ROUTE

The ROUTE statement specifies where the output of a job is to be produced. Two values may be specified; the first, ALL, PRINTER or PUNCH is optional, defaulting to PRINTER. The second specifies the location. For an up-to-date list of valid routes type HELP JD ROUTE at a console. Values at the date of printing are:-

Value	Location
CENTRAL	370/165 computer room
RECEPTION	User reception area (2 printers)
SIDGWICK	Sidgwick Avenue outstation.
WESTCAM	West Cambridge outstation
HOSPITAL	Addenbrookes' Hospital outstation
MODONE	Computer Laboratory teaching machine
ENGINEERING	Engineering Laboratory installation
CHEMISTRY	Chemical Laboratory RJE terminal
GRAPHICS	PDP 11/45 - Vector General Display
DAMTP	Press Site RJE terminal
CONTENG	Control Engineering PDP 11/45
NEWCASTLE	University of Newcastle
NORWICH	University of East Anglia
DUMMY	output abandoned
CAD	SRC PDP 11/45

Dial-up RJE terminals may also be assigned their own route names.

Examples: ROUTE NEWCASTLE

ROUTE PRINTER WESTCAM

The default ROUTE for all jobs submitted at the central site is RECEPTION. However, if the length of the output is greater than 10K lines, the output is automatically held for hand scheduling by the operations staff.

A job submitted via the internal reader (that is from a Phoenix console session or by another job) defaults to RECEPTION.

A job submitted from any other source is usually routed, by default, to its source except that if that place has no card punch (or paper-tape punch set up as a pseudo card punch) known to the system, PUNCH output is produced at CENTRAL. If ROUTE specifies either PRINTER or PUNCH only, then the other output is routed to its default destination. Printer output at RJE stations is printed in an order based on the length of the output.

TAPE7

This statement describes a 7-track tape or tapes that the job is going to use. The value is the tape name or names; if there are several names they must be separated by spaces. Each tape name may be followed by the qualifier /R or /W. /R indicates that the tape is to be read from, and may be used for any tape irrespective of its access state. /W indicates that the tape is to be written to and may only be used for READ/WRITE and WRITE ON REQUEST tapes. If no access is set, the tape is mounted in its default state. See Section 3.2.10 for further details.

Example: TAPE7 RJS101 SPQR01/R

N.B. See also the DECKS statement.

TAPE9

This statement describes a 9-track tape or tapes that a job is going to use. The value is the tape name or names. Every tape used by a job must be described by a TAPE7 or a TAPE9 statement, otherwise jobs will be cancelled automatically by the system. Each tape name may be followed by the qualifier /R or /W. /R indicates that the tape is to be read from, and may be used for any tape irrespective of its access state. /W indicates that the tape is to be written to and may only be used for READ/WRITE and WRITE ON REQUEST tapes. If no access is set, the tape is mounted in its default state. See Section 3.2.10 for further details. See also the DECKS statement.

Example: TAPE9 RJS109/W

TURNROUND

This statement enables a user to specify the maximum time that he would like the job to spend in the job queue before it is run. It may be followed by NOW or by an interval of time in minutes, hours or days, by OVERNIGHT or by a 24 hour time (<2400) preceded by BY or AT. OVERNIGHT is equivalent to BY 0800. The OR facility allows a choice of turnround. REJECT causes the job to be rejected. For a full discussion of the use of the PRIORITY and TURNROUND parameters see Appendix F. The AT option allows a job to be scheduled for a specific time, even when it would be cheaper to run the job sooner. This is only useful in connection with CHAIN,

and should rarely be used. If TURNROUND does not appear, the system inserts a statement

TURNROUND NOW OR OVERNIGHT

This causes the job to be run immediately, if the priority limit allows this, and to be run overnight otherwise.

Examples: TURNROUND 10 MINS

TURNROUND 5 MINS OR REJECT

TURNROUND BY 1200 OR OVERNIGHT

TURNROUND AT 1300

TR is a synonym for TURNROUND.

The maximum value that may be specified is 4 DAYS.

2.6 SYSTEM OUTPUT

Every job run on the 370/165 produces a certain amount of output known as 'system output'; this is normally sent to the printer, and falls into two sections, the HASP system log and the JCL reflection and messages. Any printer output generated by the job itself (e.g. compiler messages, run-time results, listings) follows the system output. At the end of the printer output from a job is a single-line trailer similar to the header (see below). RJE terminals may have different headers and trailers from the central site.

In the case of a Phoenix job, the Phoenix command reflections are not strictly part of the system output, and need not be sent to the printer; however, by default, they usually do follow the system output, and may also contain error messages.

The following information is contained in the HASP Header: the job number; the time and date that the job started to print, the POST field (see Section 2.5) and the userid under which the job was run.

2.6.1 PHOENIX AND JCL JOBS

The HASP system log which follows the header briefly describes the progress of the job. Magnetic tape mounts and demounts are recorded here, together with certain types of error message; for example:

*JOB 123 CPU TIME ESTIMATE EXCEEDED

*JOB 987 ESTIMATED LINES EXCEEDED

For an explanation of lines containing error numbers consult either 'Error Tracing and Debugging Aids' or the IBM Manual, 'Messages and Codes' (GC28-6631) e.g.

L*JOB 456 IEC209I J456ABC1 ABC101 0C0 TR=000,EG=003,CL=000,N=001,SIO=00321

records that tape ABC101 was mounted on unit 0C0 for job J456 for ABC1. For a full discussion of the message consult the entry for message IEC209I in Messages and Codes.

The HASP job statistics record the resource units charged for the job, the number of cards read in and the number of lines printed for the job, together with the number of cards (or paper tape or plotter records) punched and the length of time that the job was executing in the machine.

The JCL log begins with two lines indicating the size of the job and options requested. Note that the Cambridge job description is not reflected, though some of the parameters appear in the first two lines of the JCL log. The next line gives information about the project under which the job was run. The values listed are V, U, W, P, S, M, N. For further information about these values see Appendix F.

The next line indicates the last time that a valid job using this identifier was read into the 370/165 and the time and date that this job was submitted.

Each JCL statement is reflected, and, if the first value of the MSGLEVEL parameter is set to 1, expanded. Each step is concluded by one of the following lines:

IEF142I - STEP WAS EXECUTED - COND CODE nnn
IEF272I - STEP WAS NOT EXECUTED
COMPLETION CODE - SYSTEM = nnn USER = nnn

followed by the I/O times for disc and tape used, with the CPU time used and the amount of store if applicable. The whole job is concluded by the sum of the tape and disc times for the whole job. Note that a Phoenix job is always one job step from the JCL point of view.

Error messages which may appear in the JCL reflection are usually messages associated with APPEND. Details of these messages are given in 'Error Tracing and Debugging Aids' or in 'Messages and Codes'.

2.6.2 PHOENIX MONITOR

One of the few ways in which the user can recognise that a Phoenix job has been processed by the Phoenix Monitor is by the format of the output.

The HASP header is immediately followed by a line which gives information about the project under which the job was run. See Appendix F for details of these parameters.

The next line indicates the last time that a valid job using this identifier was read into the 370/165 and the time and date that this job was submitted. It is followed by a line which indicates the amount of cpu time used and the disc input/output time used.

The HASP job statistics record the resource units used by the job, the number of cards read in and the number of lines printed for the job, together with the number of cards (or paper tape or plotter records) punched and the length of time that the job was executing in the machine.

Output from the job follows immediately except that if a Phoenix job fails some JCL output may be produced; this can be ignored.

2.7 RESOURCE ALLOCATION AND ACCOUNTING

Every user of the 370/165 is issued with a user identifier and has access to a project number. A user identifier consists of the user's initials followed by enough digits to make a string of four characters.

Project numbers define the resource allocation to be made available to the project. Several users may have access to a single project and a user may have several projects. For accounting purposes projects are classified into groups. Each month a summary of the resources used by each project in a group is sent to the group User Representative. Detailed listings of the statistics are available for consultation in the Users' Library.

The amount of work done on a project affects the speed with which later jobs will be processed, so that it is in the interest of every user to protect his resource allocations from misuse. A password facility is provided, which allows a job to run only when the correct password is quoted with the user identifier and project number. Full details of the PASSWORD job description parameter are included in Section 2.5.

2.7.1 RESOURCE CONTROL

Resource allocation on the 370/165 is administered using the concept of 'shares' Every project has allocated to it a certain number of shares which assign to that project a proportion of the resource units available. For example, if 2000 shares are issued and a project has 10 shares, that project is entitled to 0.5% of the available resources. A description of the charging mechanism is given in Appendix F.

The way in which the scheduling system appears to the user if his project is using sufficiently small resources relative to its shares is that his small jobs receive immediate turnround at all times. As the rate of working increases, the user may find that while he may be able to get immediate turnround for a small proportion of small jobs at busy times of the day, a larger proportion or all of the project's jobs will only get immediate turnround at less busy times of the day. The user may use the PRIORITY and TURNROUND JD statements to improve the turnround of a job, but this is at the expense of subsequent jobs. Users who find their share allocation inadequate should apply via Reception to have their allocation reviewed.

Note that at the time of printing, Phoenix console sessions are unrestricted, and no charge is made against the user's resource allocation for a session.

2.7.2 RESOURCE UNITS

The resources at present taken into account in assessing the cost of a job are computation time (COMP), the amount of store that a job requires (LIMSTORE) and the number of tape decks required; a charge is also made for using private disc packs and for using the BIGDISC facility. The number of resource units used by a job are printed in the HASP job statistics (see 2.6.1). Resource units for Phoenix monitor jobs are calculated by the formula

$$\sum_{\substack{\text{all}\\ \text{commands}}} \frac{\text{store} * \text{comp}}{120K}$$

For all other jobs the formula is

$$\sum_{\substack{\text{all}\\ \text{steps}}} \left(\frac{\text{comp} * \text{store}}{120K} + \frac{1}{2} \right) + 1 + 5 * \text{no. of tape decks}$$

a charge of 5 units is also made for BIGDISC and for a private disc pack. A full discussion of the way in which the various resources are used in charging and scheduling is in Appendix F.

2.8 JOB LIMITS AND RESTRICTED FACILITIES

2.8.1 JOB LIMITS

Every job submitted to the 370/165 must include in the JD, either implicitly or explicitly, a list of the resources it requires.

Jobs exceeding their JD limits will be cancelled during execution; jobs which request resources exceeding the limits listed below will be cancelled on input with a job description error. The following limits are in force at present:

	Default	Maximum	JD parameter
real time limit	1 hour	1 hour	-
wait time limit	15 mins	15 mins	-
store size	120K bytes	400K bytes	LIMSTORE
cpu time	5 secs	20 mins	COMP
9 track tape decks	0	4	DECKS 9
7 track tape decks	0	2	DECKS 7
tapes	0	6	TAPE9 or TAPE7
printer	500	80K	PRINTER
punch (cards and paper tape) *	0	10K	PUNCH
plotter *	0	10K	PLOTTER

*Any job which uses both punch and plotter is limited to a total of 10K records. The maximum number of output streams which may be allocated to either or both of the plotter and paper tape punch is two per job.

Phoenix jobs which fall within the following limits will be run through the Phoenix Monitor:

cpu time	5 secs
store size	130K
disc I/O time	15 secs
wait limit	30 secs

no tapes, BIGDISC, private discs, plotter
no JCL statements preceding the Phoenix commands

2.8.2 RESTRICTED FACILITIES

Some parts of the system are only available to users after special authorisation by the Superintendent of Computing Services. The following facilities are at present restricted:

BIGDISC	allows the use of large amounts (>2000 tracks) of temporary disc space.
PRIVATE DISC PACKS	many of the conventions referred to in this manual do not apply to private disc packs.
WIDE PLOTTER	the width is 836 mm.
SPECIAL PLOTTER	this includes the use of standard width mylar films and inky pens.

3. DATA MANAGEMENT

This chapter describes the various ways of using datasets on disc, magnetic or paper tape and the plotter. Most of the information is independent of the control language being used (Phoenix or JCL).

The Cambridge 370/165 supports standard IBM data management with some restrictions and extensions defined below. Datasets on disc and tape whether created under JCL or Phoenix are of standard IBM form, although the IBM disc management and allocation routines have been heavily modified to provide a more flexible and efficient system. In particular, dynamic allocation has been provided and is always available both in a job and during a console session. Further details of this may be found in the publication 'Cambridge Aids to Assembler Programmers' and in Technical Report no. 4 'The Dynamic Creation of I/O Paths Under OS/360-MVT'.

3.1 MAGNETIC DISC STORAGE

Two types of disc storage device are included in the Cambridge 370/165 configuration. The first is the 3330 disc unit; the Cambridge configuration contains fourteen disc drives of this type. All the drives accept exchangeable disc packs, although four of the fourteen drives use double density packs while the other ten are single density. The disc packs each contain 13165 bytes per track and 19 tracks per cylinder, with the single density packs having around 400 cylinders per pack and the double density about twice this figure. Thus, the single density discs can hold about 100Mb of data and the double density about 200Mb. The second type of disc is the 2305 fixed head disc, a unit which is used for the swap area of the Phoenix system and for the storage of some heavily used system datasets. It is not available for the storage of user datasets.

Most of the disc units contain permanently mounted packs. Two are available to allow mounting of either private disc packs or the BIGDISC pack (see below). These are used to permit very large disc dataset creation and manipulation. The space available on the permanently mounted discs is divided between space for system use, for temporary datasets and for the filing system; this last provides facilities for the management of users' permanent datasets.

3.1.1 TEMPORARY DISC DATASETS

Temporary disc datasets may be created within a console session or job and are automatically deleted at the end of either, if they still exist. They may be of any size, although requests of above 2000 tracks (especially if made in secondary rather than primary allocations) may fail because of disc space fragmentation or more than sixteen extents being needed. If this failure occurs the BIGDISC facility may be needed.

When using JCL the standard IBM convention for temporary dataset names

is used, that is one or two ampersands followed by a simple-name which is between one and eight characters long, the first character being an alphabetic or national character and any following being alphanumerics. The form with two ampersands is to be preferred to prevent confusion with JCL symbolic parameters. In JCL the dataset names &T and &&T refer to the same dataset; however, in the Phoenix language they refer to two different datasets. Note that the Phoenix dataset name &T is not the same (and does not refer to the same dataset) as the JCL dataset name &T.

EXAMPLES

```
&T
&&A
&A123B456
```

3.1.2 THE FILING SYSTEM

The filing system controls the amount of permanent disc storage available to a user, while attempting to give him maximum flexibility in organising his own use of space. A permanent (or catalogued) disc dataset is one that is created within a job or console session, and that stays available for use in subsequent jobs or console sessions until explicitly deleted. All permanent datasets must belong to a fileowner who has been allocated a fileowner name (usually the same as the userid) by the administration, and they must follow the naming convention:

```
Fileowner.simple-name
or
Fileowner.simple-name.simple-name
```

That is, two or three components separated by periods: the first component being the fileowner name and the second and third simple-names, see 3.1.1. Thus sample dataset names are:

```
RJS9.ABCDEFGH.IJKLMNOP
KE14.Z
PS16.VER3.MAY75
```

There is an OS restriction that no three-component dataset name may be used with the same first two components as an existing two-component dataset name (or vice versa).

Under Phoenix the fileowner name, or the fileowner name plus the second component, may be omitted when referring to the dataset. See the SET command for further details (Part II).

3.1.3 PERMANENT DISC SPACE CONTROL

When a user applies to be allowed to use permanent datasets on disc, he is issued with a fileowner name (usually the same as his userid) and

some parameters to control his use of disc space. These parameters are the disc limit, disc income and credit bound, which are used to limit the total permanent disc space occupied by a fileowner at any time and also to control the average amount of permanent disc space used. The units of these parameters are tracks and a typical allocation for a fileowner might be disc limit (L) 20 tracks, income (I) 10 tracks and a credit bound (B) of 30 tracks. This will allow the fileowner to have permanent datasets occupying a maximum of 20 tracks at any one time, as long as on average he keeps 10 tracks or less on disc.

Control of the average is achieved by use of a credit system, in which all fileowners have stored in a database (called JOB0) a value in tracks called their current credit (C). At approximately 9.00 a.m. every day (including non-working days) each credit is adjusted by adding the amount given by the fileowner's income and subtracting the amount of permanent disc space he is using at that time. The balance is always kept at or below the credit bound and at or above a lower limit of minus the disc limit. The purpose of the credit bound is to stop a fileowner's credit from rising too high when he is not computing. The lower limit makes sure that a fileowner is not too heavily penalised for accidentally letting his balance become negative.

A negative balance causes the fileowner to be inhibited, that is, he can no longer create new permanent datasets or extend old ones until his balance becomes positive, a state which he can achieve by deleting datasets or moving them to a magnetic tape or to another fileowner obtaining his permission if necessary, and then waiting for sufficient time for his usage to become positive.

These rules may be summarised as follows. If U is the disc usage and, as defined above, L is disc limit, C is credit, I is the disc income and B is the credit bound

At all times: $U \leq L$

At about 9.00 a.m. every day

$$C = MAX(-L, MIN(C-U+I, B)).$$

Users should note that the value U is the sum of the amounts of disc space allocated by the system to each dataset (and therefore is an integral number of tracks) and is not the amount actually used within the allocation.

The user's current credit and other disc control parameters can be checked using the EXAMINE command, see Part II. Details of all disc control parameters may be found in Example Sheet 27.

3.1.4 SAFETY AND SECURITY OF DATASETS

The IBM dataset protection facilities of password, expiry date and retention date protection are not available to users on the Cambridge 370/165. They have been withdrawn because their use would cause an unacceptable load on the operations staff and a degradation of the overall service.

There is no automatic archiving of user datasets. Although there is a weekly total dump of the system, it is almost impossible to retrieve a small number of datasets. The dump is used to provide security against a major system collapse. It is therefore the individual user's responsibility to safeguard his own datasets by keeping copies on disc or tape. In particular archiving by the use of TLS is recommended, further details of which may be found in Part II.

3.1.5 PRIVATE DISCS

In some circumstances a user may need the speed of access of disc in combination with the storage capacity of magnetic tape. For example, this need might arise in the development of a large database and it may be appropriate to consider the use of a private disc pack. These are normally owned by a user or group and their administration and contents are in the control of that user or group. Note that datasets on a private pack may not be catalogued.

Any job that uses a private pack must contain the JD parameter DISC followed by the disc pack name which is normally a six character identifier. Private disc packs may be used in Phoenix jobs, but they must be defined by a JCL dd statement (preceding the Phoenix commands). Private disc packs must be identified using the parameters

VOL=SER=pack name and UNIT=DISC

(note that the first may only be used with the private disc facility). When a dataset is created the DISP subparameter KEEP should be used e.g. DISP=(,KEEP).

The use of the private pack facility entails a scheduling charge, details of which are given in Chapter 2. Private discs are not available within Phoenix Monitor jobs.

Application for use of a private disc pack should be made to the Superintendent of Computing Services.

EXAMPLES

(a)
```
JOB SPQR 999
DISC PRIV00
PHOENIX
//A DD DSN=A.B,VOL=SER=PRIV00,UNIT=DISC,DISP=SHR
: .....
```
Use in a Phoenix job.

(b)
```
JOB SPQR 999
DISC PRIV00
JCL
// EXEC .....
//FROM DD DSN=C.D,UNIT=DISC,VOL=SER=PRIV00,DISP=SHR
```
Use in a JCL job.

3.1.6 BIGDISC

Users who find that they cannot get enough temporary disc space through the normal request procedure in a JCL or Phoenix job may be authorised to use the BIGDISC facility. This allows the use of a complete disc pack. To use this, the job description must contain the parameter BIGDISC and any dd statement that defines a large temporary dataset should contain the parameter VOL=REF=BIGDISC.

The facility may be used in a Phoenix job but the dataset has to be defined by a JCL dd statement (preceding the Phoenix commands); there is no way to refer to BIGDISC in the Phoenix language. BIGDISC may not be used for permanent datasets and indeed all datasets on the BIGDISC pack are automatically deleted at the end of the job. Note that an extra scheduling charge is made for this facility; see Chapter 2 for details. BIGDISC is not available in Phoenix Monitor jobs.

Application for the use of this facility should be made to Reception.

EXAMPLES

(a)
```
JOB XYZ1 2345
BIGDISC
PHOENIX
//A DD DSN=&&A,VOL=REF=BIGDISC,
// SPACE=(TRK,(300,100))
C .....
```

Use in a Phoenix job.

(b)
```
JOB XYZ1 2345
BIGDISC
JCL
// EXEC .....
//LARGE DD DSN=&&LA,VOL=REF=BIGDISC,
// SPACE=(CYL,(20,10))
```

Use in a JCL job.

3.2 MAGNETIC TAPES

3.2.1 HARDWARE

The Cambridge 370/165 has eight 9-track 3420-3 Magnetic Tape drives. These are cartridge loading, 75 inches per second drives accepting industry compatible 1/2 inch tape. 9-track tape can be written in two densities, either 800 bpi or 1600 bpi (bpi=bits per inch). Densities must not be mixed on the same tape. 9-track tapes are always written in odd parity. Initialisation of the volume label on a tape is at 1600 bpi unless 800 bpi is specifically requested by the owner.

There are also two 7-track 3420-3 tape drives. The density on 7-track tape may be 556 bpi or 800 bpi and initial labelling is at 800 bpi unless 556 bpi is specifically requested. 7-track tape should only be used for tapes which come from or are to be sent to another installation.

3.2.2 ADMINISTRATION OF MAGNETIC TAPES

Tapes are issued to a user on application to Computing Service Reception on a free loan basis (although if large numbers of tapes are used, some cost sharing may be necessary). Tapes are normally stored in the 370 machine room and are mounted by the operations staff when a job requests them. Tapes may be removed from the Computing Service only after application to the Magnetic Tape Librarian Tapes which are brought to the Computing Service must be registered with the Tape Librarian.

3.2.3 USE OF MAGNETIC TAPE

There are two general methods of using magnetic tape on the Cambridge 370/165, one of which is the standard IBM method and the other a Cambridge development called the Tape Library System (TLS). TLS uses a directory at the beginning of the tape which contains information about the contents of the tape, thus allowing retrieval of datasets by the name alone. The two methods may not be mixed on one tape, although sequential datasets which have been written to tape using TLS may be read using IBM standard methods Note that partitioned datasets which have been written to tape using TLS cannot be read by any other program.

3.2.4 NAMES AND LABELS

Magnetic tapes are normally issued initialised with IBM standard labels; they may, in addition and on request, be initialised for TLS. Unlabelled tapes should only be used to transfer information to and from machines that use non-standard labels They may not be used with TLS.

Standard labelled tapes are normally given names consisting of six characters, the first four of which are the userid of the owner and the last pair are numerals chosen by the owner. Unlabelled tapes have similar names except that they start with the two numerals and continue with the userid

3.2.5 JOBS

The Phoenix command language cannot refer to magnetic tape datasets directly but must access them via JCL dd statements that follow the PHOENIX job description parameter and precede the command sequence.

Note that Phoenix Monitor jobs may not access magnetic tapes at all.

JOB DESCRIPTION PARAMETERS

All magnetic tape jobs require the TAPE7 or TAPE9 parameter which defines the names of the tapes to be used. In addition, in a multi-step JCL job in which the tapes are used sequentially rather than in parallel, the DECKS parameter should be used Further details of these parameters may be found in Section 2.5.

3.2.6 TAPE PARAMETERS ON DATA DEFINITION STATEMENTS

Full details of the following parameters are included in the IBM publication JCL Reference Manual (GC28-6704).

THE LABEL PARAMETER

BLP (Bypass Label Processing) may not be used, except after consultation with the Computing Service, as it is liable to cause overwriting of random tapes. All tapes are initialised by the Computing Service before they are issued so that users only need to use BLP under exceptional circumstances.

If a tape is used without a write-permit ring, IN should be coded e g.

```
LABEL=(4,SL,,IN)
```

PASSWORD, NOPWREAD, EXPDT, or RETPD may not be used in the LABEL field.

THE DCB SUBPARAMETER DEN

It should be noted that all datasets on the same magnetic tape must be written with the same density; otherwise they will not be able to be read.

The DEN sub-parameter of the DCB parameter may have values as in the following table:

DEN	7-track	9-track
1	556	
2	800	800
3		1600

The default for 7-track tape is 800 bpi and for 9-track 1600 bpi.

THE DCB SUBPARAMETER TRTCH FOR 7-TRACK TAPE

Data should be written to 7-track tape using either the data conversion feature or the translation feature. If the translation feature is used, users are advised to record in odd parity to avoid loss of information.

It should be noted that the default for the TRTCH DCP subparameter is odd parity and no translation or data conversion; thus, on writing, the two most significant bits of each byte are lost This default should clearly be avoided.

See the JCL Reference Manual for details of the TRTCH subparameter.

THE UNIT PARAMETER

The group names used as values of the UNIT parameter should be TAPE7 for 7-track tapes and TAPE9 for 9-track tapes.

The unit count or P parameter should not normally be used Users should consult the Computing Service if they believe that they could profitably use this facility. As a general rule, multivolume tape datasets should only use more than one unit if used in a jobstep occupying more than 250K of core and then the unit count should not be greater than 2 If this facility is used, the volume count parameter must be correct in the VOL field

Deferred mounting (DEFER) may not be used

THE VOL PARAMETER

PRIVATE must be used if no dataset name is quoted It is preferable always to use PRIVATE.

RETAIN must be used if the same tape is used for more than one dataset in the same job, or if datasets on the same tape are concatenated or if the same dataset is opened more than once

RETAIN should not normally be coded for the last step of a job unless a concatenated tape dataset is used or the same dataset is opened more than once in that step

The volume count field should always be used for multivolume datasets and also when writing to a dataset near the end of a tape Multivolume datasets should not normally be used (see above)

See the JCL Reference Manual (GC28-6704) for details of the VOL field

THE DSN PARAMETER

This field is limited to 17 characters If a name of more than 17 characters is given, then the extra leading characters are removed The DSN parameter must always be used for labelled tapes; otherwise the system will invent a name and the user will not then be able easily to retrieve the dataset The invented name is of the form Jnnnuserid R000000m where nnn is the job number and the dataset is the mth of this type created in the jobstep

It is good practice to use a dataset name even for unlabelled tapes If a dataset name is not used for unlabelled tapes the PRIVATE subparameter of the VOL parameter must be coded

A job that does not conform to the above rules may cause other jobs to be held up or may even cause the system to wait - for example an operator reply might be needed Such a job may be cancelled by the operator

3 2 7 TAPE MARKS

On 9-track 1600 bpi tape the interblock gap is followed by between twenty and sixty-four identical characters consisting of ones in the P,0,5, or 2 6,7 positions and no signal in the 1,3,4 positions

On 9-track 800 bpi tape the interblock gap is followed by the tape mark defined by ones in the 3, 6 and 7 positions, and a following identical check character, 8 bit positions later.

Bit positions across 9-track tape are numbered as follows: 4,6,0,1,2,P, 3,7,5.

End of file on 7-track tape is defined as an interblock gap followed by a tapemark (001111) and a following identical check character, 4 bit positions later.

3.2.8 SCRATCH TAPES

There are certain tapes which may be used by users as scratch tapes within a job. Information written to them is deemed to be lost at the end of a job. They have been initialised with standard labels.

There are the following four 9-track tapes written at 1600 bpi

USER91,USER92,USER93,USER94

and two 7-track tapes written at 800 bpi which are

USER71,USER72

3.2.9 TRANSFER OF MAGNETIC TAPES

The specifications below are recommended for transfer of magnetic tapes from other installations.

The standard specification for 7-track tape is:

Density :	556 bpi
Parity :	Odd
Character code :	BCD
Block size :	800 characters, each block consisting of 10 records (card images) of 80 characters each
Inter-block gap :	Long (0.68 inch minimum)

For a 9-track tape, the normal options would be :

Density :	800 bpi
Parity :	Odd
Character code :	EBCDIC
Block size :	800 characters as above
Inter-block gap :	0.5 inch minimum (this is standard)
Label :	none

Alternatives which can be read at Cambridge are :

Density :	800 bpi for 7-track or 1600 bpi (phase-encoded) for 9-track
Parity :	may be even for 7-track

Various character codes (consult the Computing Service for details).

Various block sizes (minimum 18 characters); records need not be card images, but variable-length records may give rise to considerable disentangling problems.

The long inter-block gap on 7-track tape is essential; tapes with short gaps cannot be read on the Cambridge decks.

9-track tapes may have standard OS labels, but other types of label may cause problems; unlabelled tapes are preferred. Standard labels should, if possible, conform to the Cambridge naming conventions.

If a tape to be read does not conform to the standard specification, it is essential that details of its blocking, character code, etc. be supplied by the user.

All tapes coming from outside must be registered with the Tape Librarian.

Note also that the 3420-3 tape decks are automatic threading ones. This means that tapes with rough ends may need a few inches of tape trimmed off in order to load successfully. For single transfers this normally will not matter as there are usually a few feet of lead-in tape, but may be important if the process is repeated.

3.2.10 MAGNETIC TAPE WRITE-PERMIT RINGS

All magnetic tapes on the Cambridge 370/165 have one of three access states, two of which may be overridden by coding the qualifier /R or /W on the TAPE9 or TAPE7 job description statement. The three states are:-

(a)	READ/WRITE (ring IN)	By default, the write-permit ring is in the tape when it is mounted, and the tape may be read from or written to. No qualifier (/W or /R) needs to be quoted, but if /R is specified the tape may not be opened for output or INOUT. (See note below).
(b)	WRITE ON REQUEST (ring OUT)	The write-permit ring is not normally in the tape, and the tape may not be written to, but may always be read (see note below); if /R is specified in the JD it has no effect. The tape may only be written to when /W is specified in the JD.

(c)	READ ONLY (ring NEVER IN)	The write-permit ring is never in the tape, and the tape may never be written to. Any attempt to open the tape for output, by coding /W in the JD will cause the job to be cancelled at runtime with code S222.

N.B. Any attempt to open a tape without a write-permit ring for OUTPUT or for INOUT will cause the job to abend with code S613 (IEC147I 613-0C). Users who wish to read from a dataset on magnetic tape, but are using a system which opens datasets for INOUT (e.g. Fortran) must code LABEL=(,,,IN) on their tape dd statement.

New tapes are issued with READ/WRITE access unless the user requests otherwise. The access state of a tape may be altered by contacting the Magnetic Tape Librarian. The Phoenix command and JCL catalogued procedure TAPESTAT are provided for users to find out the access states of their tapes (see Part II).

3.3 DCB PARAMETERS, SIZE PARAMETERS AND ACCESS METHODS

3.3.1 ACCESS METHODS AND DATASET TYPES

Sequential (PS), direct access (DA) and partitioned datasets (PO) are supported. Indexed sequential datasets are not. Note also that the FBS dataset format must not be used with partitioned datasets.

BSAM, QSAM, BPAM and BDAM are supported. ISAM is not supported. In addition the Cambridge module called NIOP (New Input Output Package) is available and will deal with most sequential access dataset formats. NIOP is fully defined in the publication 'Cambridge Aids to Assembler Programmers'.

3.3.2 BLOCKING FACTORS

Users are reminded that the inter record gap (i.e. the space between physical records) is of the order of .75 inches on magnetic tape (i.e. around 600 bytes for 800 bpi) and of the order of 135 bytes on disc.

It is therefore important for efficient use of discs and magnetic tape to make the physical record size (BLKSIZE) as large as is consistent with minimising the wastage from unfilled final blocks and keeping the size of input buffers small.

A 3336-1 disc pack contains about 400 cylinders and a 3336-11 about 800 cylinders. Each cylinder contains 19 tracks and each track contains 13165 bytes.

The maximum number of blocks per track on a 3336 disc pack is given by the formula

$$n = \frac{13165}{135+B}$$ (no keys) where B is the block size

The maximum block size or number of bytes available per track for various values of n, and various formats are

n	VBS,VB,V or U BLKSIZE	bytes/track	FB,LRECL=80 BLKSIZE	bytes/track
1	13030	13030	12960	12960
2	6447	12894	6400	12800
3	4253	12759	4240	12740
4	3156	12624	3120	12480
5	2498	12490	2480	12400
6	2059	12354	2000	12000
7	1745	12215	1680	11760
8	1510	12080	1440	11520
9	1327	11943	1280	11520
10	1181	11810	1120	11200

3.3.3 SIMPLIFIED DCB STATEMENTS FOR JCL

Advantage has been taken of a facility in JCL that allows DCB characteristics to be picked up by reference rather than explicitly, so that a simplified DCB parameter can be used for a number of standard formats for the creation of datasets on disc or tape. These formats may also be used as qualifiers in Phoenix dataset descriptions. (See Section 4.3.3). The DCB parameter will only be used if the disposition is new or if the dataset to which it refers is a SYSOUT dataset.

The simplified form is DCB=value where the possible values are given below.

Value	Equivalent Full Format		
	RECFM	LRECL	BLKSIZE
FA	FA	133	133
FB	FB	80	2480
FBA	FBA	133	1995
FBS	FBS	80	2480
FF	F	80	80
FP	FA	121	121

VB	VB	84	2498
VBA	VBA	137	2498
VBL	VB	136	2498
VBS	VBS	32756	2498
VBSL	VBS	136	2498
VBST	VBS	32756	13030
VBX	VBS	X	2498

Note that VBL is the format recommended for efficient storage on disc of any but fixed record length data. FP, FBA and VBA are useful for printer datasets and FB for temporary datasets that are to be read by most IBM Fortran compilers.

EXAMPLES

Phoenix

(a)
```
FILE TO .A/VBA
```

(b)
```
FILE TO &A/FB FROM .B:C
FTG1CLG PROGRAM &A
```

JCL

(c)
```
// EXEC FILE
//FROM DD *
.....
//TO DD DSN=SPQR.A,DISP=(,CATLG),UNIT=DISC,
//    SPACE=(TRK,(1,1)),DCB=VBL
```

(d)
```
// EXEC FILE
//FROM DD DSN=SPQR.A,DISP=SHR
//TO DD DSN=&&A,DISP=(,PASS),UNIT=DISC,
//   SPACE=(TRK,(1,1)),DCB=FB
// EXEC FTG1CLG
//FORT.SYSIN DD DSN=&&A,DISP=(OLD,DELETE)
```

3.3.4 PHOENIX SIZE QUALIFIERS

The size of a dataset created under Phoenix may be specified by the qualifiers TRK and CYL or globally using the SET SPACE command (see SPACE in Part II).

The format of the TRK and CYL qualifiers is

TRKp,q,r and CYLp,q,r

where p defines the primary allocation, q the secondary allocation in tracks or cylinders and r defines the number of directory blocks. If any of p, q or r are omitted they are given the value zero. Cylinder allocations should not normally be used.

There is also a set of size qualifiers which are easier to use than the above; these are:-

	Primary (tracks)	Secondary (tracks)	Directory (blocks)
SMALL) S)	1	1	3
NORMAL) NL)	2	2	5
MEDIUM) MED)	2	5	5
LARGE) LA)	10	20	10
GIGANTIC) GIGA) G)	20	50	200

3.4 PAPER TAPE

3.4.1 PAPER TAPE INPUT

Paper tape input and output is provided centrally via the PDP11 communications system and two TREND paper tape reader/punches. Input from the PDP11 to the 370/165 simulates a card reader, that is to say, complete 80 byte records are always handed to the 370/165.

The Sidgwick and Chemistry RJE terminals support paper tape input with the same specification.

3.4.2 INPUT WIDTHS AND CODES

5, 7 and 8 track tape may be input. 5 track may be input as binary only, 7 track as Titan flexowriter code or as binary, 8 track as odd, even or non parity ASCII or as binary.

Character codes are translated into EBCDIC (using the augmented character set as defined in Appendix A). Characters which are not assigned in EBCDIC are faulted. For 7 track flexowriter code the characters below are translated as follows:

§	becomes	$
→	becomes	←
∿	becomes	¬
χ	becomes	@

To input characters that are undefined in the particular code used, the escape character @ or Flexowriter χ may be used, this being followed by one or more escaped characters as defined in Appendix B. (But see options below)

Each paper tape presented to the machine must either contain a complete job or it must be one of a series of tapes which may be terminated with a Cartrifile and which make up a complete job consisting of a valid Job Description followed by JCL statements or Phoenix commands

3.4.3 ERROR MESSAGES

The following information is given in the case of an error during the reading of a job. The operator will copy the error message onto a comment slip and return it with the job. Possible messages are:-

NO WS	no warning sequence, i.e. unterminated translated tape.
BAD WS	'+++' at the start of a line followed by an unknown character.
FAULT CHAR n	Unassigned character, OCTAL value n.
BAD ESCAPE n	'@' followed by an unknown character, OCTAL value n, or '@X' not followed by two hexadecimal digits

If the error occurred in the middle of a tape, the operator will mark a point on the tape (within a few characters) after the point at which the fault character occurred.

3.4.4 WARNING SEQUENCES

Warning sequences are used to signal to the machine some change in the tape or the translation option to be used, (see OPTIONS below) Upper and lower case letters are synonymous in warning sequences Warning sequences are only recognised at the start of a record This record is not transmitted to the 370/165. For those warning sequences which do not end the tape (B,P,S) the rest of the record is ignored. Possible warning sequences are:

++++	End of job and physical tape.
+++Z	End of job and physical tape.
+++T	End of physical tape, job continues on next tape.

+++. Suppress fault characters for the rest of the job.

+++F End of physical tape, job continues in binary on next tape.

+++B Job continues in binary on the same tape after the next end of record character

+++E End of physical tape job continues on next tape in translated binary mode (see below)

+++Ø Change warning character from + to Ø Ø may be any printing, character other than a letter or a digit

+++S Job continues on same tape in stream mode see below)

+++K End of physical tape job continues on 16 bit Cartrifile

+++L End of physical tape job continues on 8 bit Cartrifile

After +++F, +++B and +++F the binary tape is terminated when it runs out of the reader. The final record is padded with binary zeros It is wise to terminate these tapes with at least one foot of run-out Unassigned or incorrect parity characters and unknown escape combinations following +++P are replaced by X'FF' i e the hexadecimal number FF.

3 4 5 OPTIONS

Four translation options are provided for reading paper tape

a) Normal (translated) mode

This mode is set at the start of each job. Characters are translated from ASCII or flexowriter code to EBCDIC Card images are formed by filling out short lines with spaces, and by splitting up long lines into 80-byte segments Carriage control characters are not present in the transmitted records. The character erase is totally ignored Tab characters are expanded to the appropriate number of spaces up to the next tab position Tab positions are every 8 characters For flexowriter 7 track tape at least two spaces are generated (double space tab, while for ASCII 8 track it is at least one (single space tab) Escape combinations are recognised; in particular, @C and @L are available for local line editing, but note that they only apply within the current 80-byte segment of long lines, and that @C following a tab will only cancel one space Warning sequences are recognised and a warning sequence is required at the end of the tape. Since erase is ignored a string such as +++ erase>Z will be recognised as a warning sequence If it is desired to input three +'s at the start of a line, @N+++' or @C+++ can be used.

(b) Stream (translated) mode

This mode is introduced by the warning sequence +++S (which does not terminate the physical tape). Characters are translated to EBCDIC as in normal mode. However, card images are not created by padding or splitting lines; instead, the end of each logical line is indicated by the character newline (X'15'). The next logical line follows immediately in the same card image; physical record boundaries are therefore meaningless. The characters backspace, erase, and tab are translated to their EBCDIC equivalents (X'16', X'07', X'05'), and no interpretation of escape combinations takes place. Warning sequences are still recognised (at the start of a new logical record, as always), and when a warning sequence is encountered, the current physical record (card image) is padded out with the NUL character (X'00') and transmitted. As erases are not ignored in this mode, a string such as '+++<erase>Z' will NOT be recognised as a warning sequence.

The procedure IMAGE (see Part II) has an option STREAM which is specifically designed to accept data in the above format, line-image it, interpret escapes, and re-create the logical records.

(c) Translated binary mode

This mode is introduced by the warning sequence +++E, which also terminates a physical tape. It is intended for reading in tapes produced elsewhere which do not conform to the local conventions. Translation, padding, and splitting are carried out as in normal mode; however, escape combinations are not recognised, nor are warning sequences. Tabs are still expanded, erases ignored, and backspaces and unassigned characters faulted. The tape is read until it runs out of the reader.

(d) Binary mode

This mode is introduced by the warning sequence +++B (which does not terminate the physical tape) or +++F (which does). Card images are created with each byte corresponding to a row on the tape, which is read until it runs out of the reader. Binary rows are placed right justified and padded with zeroes in eight bit bytes. Multiple binary tapes are permitted.

NOTES

(a) Once the mode has been changed from normal to any other mode, it remains changed until the end of the job.

(b) In all translated modes, unassigned characters are faulted and the job is cancelled, unless the +++P warning sequence has occurred in which case the character X'FF' is substituted. In normal mode this also applies to unassigned escape combinations.

3.4.6 JOB SUBMISSION

When submitting paper tape to the 370/165 the following rules should be followed:

(a) each tape should be labelled clearly with the user identifier.

(b) if there is more than one tape, each one should be numbered and the green job submission card should indicate the order in which the tapes should be read in.

(c) if any tapes contain information in binary it should be stated which they are.

3.4.7 PAPER TAPE OUTPUT

Paper tape output is available in a variety of tape widths and codes and also in binary. Trailing spaces are removed from non-binary tapes. Non-binary records are normally followed by carriage return and linefeed or a newline, but this can be suppressed.

RESTRICTIONS

Spanned records are not supported.

The total number of paper tape output and plotter dd statements may not exceed 2 per job.

The maximum blocksize supported is 254.

3.4.8 OUTPUT CODES AND WIDTHS

The following are available:

8-track even parity Teletype-compatible ASCII and binary
7-track Titan flexowriter code and binary
5-track binary

Other codes may be provided in the future depending on demand.

3.4.9 JOBS

A PUNCH job description parameter must be provided indicating the maximum number of records to be produced.

JOB CONTROL LANGUAGE

In the dd statement, SYSOUT=5,7 or 8 should be used with the digit indicating the width required.

DCB=MODE=C specifies binary output. Alternatively assembly code programmers may set this directly in the DCB if binary output is required. If no MODE is coded, or if DCB=MODE=E is coded, then the characters are translated to the appropriate tape code.

DCB=STACK=2 may be coded if the newline (or carriage return - line feed) following each non-binary record is to be suppressed. DCB=STACK=1 is assumed in default and causes each non-binary record to be followed by a newline or a carriage return line feed. Assembly code programmers may set STACK in the DCB as required. Alternatively, the new line in the previous record may be suppressed by writing records with ANSI carriage control character 'W'.

3.5 PERIPHERAL DEVICES

3.5.1 PLOTTER

Hardcopy graphical output may be produced directly on the Calcomp 1136 drum plotter which is connected to the 370/165 via a Calcomp 900 controller. The plotter has three pens (normally red, black and green) which may be selected by program, and can be loaded with either narrow (253 mm) or wide (836 mm) paper. The narrow paper may be used by any user but the wide paper needs special authorisation from the Superintendent of Computing Services or from the Operations Supervisor; special paper (e.g. Mylar film) or special pens (e.g. ink rather than ball point) also require special authorisation.

Note that the total number of paper tape output and plotter dd statements may not exceed 2 per job. The total number of plotter, paper tape and card records produced may not exceed 10K.

Plots may be examined on a graphical terminal before they are sent to the plotter using the VIEW command (see Part II). To use this facility plotter output should be sent to a file which has DCB=FB. The plot may be sent to the plotter later if hard copy is required. A DEC GT40 is available in the User Area for general use of this facility.

A full specification for the use of the plotter may be found in the Graphics manual.

3.5.2 PRINTERS

The central printers are IBM 1403.N1 1100 lpm train printers with a large character set. Appendix C gives details of the character set together with advice on the efficient use of the printers.

Sidgwick Avenue outstation has an MDS 4340 600 lpm chain printer with the same character set as the central site. The West Cambridge outstation has two Data Products 2260 drum printers with a 96 character set. These have a nominal speed of 436 lpm. The Addenbrookes outstation also has a Data Products 2260 printer with the same specification as the West Cambridge printers.

4. PHOENIX LANGUAGE

This chapter fully describes the Phoenix command language. It is intended mainly for those who need to construct their own libraries of command sequences or who wish to have a complete understanding of the language. Many users will find it sufficient to refer to 'Basic Use of the Cambridge 370/165' for information on how to write commands generally, and to Part II (Specifications) of this manual for how to access the various utilities and subsystems available. The chapter is in four parts - the basic syntax (sections 4.1 to 4.4), the construction and properties of datasets set up by the Phoenix system (4.5 to 4.6), macros and procedures which make use of the basic features (4.7 to 4.9), and finally various control features (4.10 and 4.11).

DEFINITIONS OF BASIC SYNTACTIC ELEMENTS

Items in italics in the text below are syntactic elements.

Square brackets around any string of characters in the rest of this chapter signify that the string enclosed is optional. Items in vertical columns are alternatives.

letter is one of a to z or A to Z.
digit is one of 0 to 9
alphanat is a *letter*, #, $, &, @, ! or £.

word is a string of up to 8 *letters*.
int is a string of up to 8 *digits*.
name is a string up to 8 characters long the first of which must be an *alphanat* and the remainder can be any of *alphanat* or *digit* or -.

In the above syntactic elements Phoenix replaces all lower case *letters* by their upper case equivalents.

text is a string composed of any sequence of characters except space or newline and is terminated by space or newline.
quoted-text is a string which is delimited by single quotes and which contains any characters except newline. Any quote inside the string is written as two quotes.
rest is a string terminated by newline.

Special characters.

The characters < > ' may be changed by use of the SET CHAR command as described in 4.11.3 and Part II (Specifications).

Unless otherwise stated, items may always be separated by single or multiple spaces.

4.1 SYNTAX OF PHOENIX BASIC COMMANDS

There are two types of basic command; control commands which are built into the Phoenix system, and program execution commands which enable arbitrary load modules to be loaded and entered with any requisite data definitions and parameter data.

The normal decoding unit is a line (i.e. a sequence of characters ending with a newline); however if the last character of a line is the continuation-character defined by the SET CHAR END command (see 4.11.3) then the decoding unit is the concatenation of the two lines.

4.1.1 CONTROL COMMANDS

These are listed in 4.10.3 below. They consist of a line starting with a *name*, *%name* or * and followed by the requisite data.

Examples:-

```
A
QUIT
%FINISH
WAIT 3
*ABC
```

The most important control command is the C command, see 4.8.1 below.

Any line starting with a *name* is a control command, a macro command (see 4.7) or a procedure call (see 4.8.2).

4.1.2 PROGRAM EXECUTION COMMANDS

The purpose of the program execution command is to provide an interface to a program which is stored as a load module. The interface provides, compatibly with the JCL interface, a parameter field, data definitions and other miscellaneous data.

The program execution command consists of up to three parts delimited by brackets and terminated by newline:-

[*command-source*] [*command-environment*] [*command-parameter*]

where the *command-environment* starts with a (and ends with a).

COMMAND-SOURCE

This names the command, and can take a number of forms.

(a)	:*name*	The command is *name* and is to be found in the system link library, or in a step or job library.

(b)	(*name*)	is the same as :*name*.
(c)	*dsname:name*	The command is *name* and is to be found in the library called *dsname*. (For definition of *dsname* see below).
(d)	*dsname(name)*	is the same as *dsname:name*.
(e)	empty	No command is entered, but allocations in the environment are carried out. The *command-source* is empty if the first significant character on the line is a (followed by any non-*alphanat* character or (followed by a *name* which is not terminated by).
(f)	other	For lines not covered by the above types see the sections on macros, procedures, and control commands.

COMMAND-ENVIRONMENT is defined in 4.2.

COMMAND-PARAMETER

This provides a piece of *text* which is passed to the command in its PARM field. The format and significance of the *text* depend on the command. (see also 4.2.3)

4.2 SYNTAX OF *COMMAND-ENVIRONMENTS*

These provide the environment in which the command runs and consist of a sequence of *environment-elements*. The basic form of an *environment-element* is A=B where A names an item and B provides a value for it. The = is optional and may be replaced by or surrounded by one or more spaces. *Environment-elements* may be separated by spaces, by one or more commas, or by newlines (but see below).

If no *environment-elements* are to be specified then the entire *command-environment* including the brackets may be omitted.

A *command-environment* may be spread over many lines, in which case the opening bracket must occur on the first line, and the closing bracket on the last line. A *command-environment* may also include several separately bracketed sections in which case nothing other than spaces may occur between the closing bracket of one section and the opening bracket of the next.

Examples:-

```
(a)     (A=B)
(b)     ( A =  B ) (C= D)
(c)     (A=B
         C=D)
(d)     (A  B  C  D )
(e)     (A B,,C =D )
```

Examples (b)-(e) are equivalent.

The classes of *environment-elements* are :-

Store requirement	(see 4.2.1)
Time requirement	(see 4.2.2)
Parameter data	(see 4.2.3)
Default settings	(see 4.2.4)
Data definitions	(see 4.3)

4.2.1 STORE REQUIREMENT

%S [=] *n*[K]

(Synonym: %REGION)

The *name* of the item is %S; the value *n* is an integer specifying the number of K bytes required. The K is optional. If %S is not included in the command, the entire available region is used.

4.2.2 TIME REQUIREMENT

%T [=] *n*[S]

(Synonym: %TIME)

The *name* of the item is %T. The value *n* is an integer specifying the cpu time required in seconds. The S is optional. If %T is not included or is equal to 0, the time for the command is only restricted by the time for the console session or job-step.

4.2.3 PARAMETER DATA

%D [=] *text*
quoted-text

(Synonym: %PARM)

The *name* of the item is %D. The value is the string provided by *text* or *quoted-text*.

This provides an alternative means to *command-parameter* for passing a parameter string to a command. If there is more than one parameter string (i.e. any mixture of one *command-parameter* and one or more %Ds) they are concatenated in order from left to right subject to an overall maximum of 255 characters.

4.2.4 DEFAULT SETTINGS

%DEF [=] *qlist*
/qlist

For definition of *qlist* see below.

This provides a means of varying the default values for the characteristics of a dataset by specifying the new defaults in the *qlist* (see 4.3.1).

4.3 DATA DEFINITIONS

Data definitions provide the means of associating *ddnames* with datasets which may be disc, terminal, dummy, printer or card punch datasets. Other devices may be specified only indirectly (via JCL), although it will be possible to specify them directly at some future date.

ddname [=] *dsd*		sets up one data definition with the *ddname* describing the dataset defined by the *dsd*.
ddseq [=] *dsdlist*		sets up as many data definitions as there are *dsd*'s in the *dsdlist*. The *ddnames* allocated are the set obtained by replacing the character : in *ddseq* by 1,2,3....9 one at a time.

4.3.1 SYNTACTIC ELEMENTS

The following are syntactic elements used in data definitions.

ddname	*name*
ddseq	A *name* with any one character other than the first replaced by a colon.
dsname	(a) has the form *name.name.name* The first *name* and the final *.name* may be omitted. This is the standard Cambridge form of the name of a catalogued dataset.

(b) *ddname*
(c) *&name*
(d) *%name*
(e) *[char]
where possible values of char are defined below.

(a)-(d) may be followed by (*name*) or :*name* indicating a pds member name.

qualifier-terminator space) / or newline.

qualifier An item starting with a character other than space) or newline, continuing with any characters and terminating with a *qualifier-terminator*.

qlist *qualifier* or *qlist/qualifier*
i.e. any number of *qualifiers* separated only by slashes.

qds *dsname* or *dsname/qlist*
i.e. *dsname* followed by any number of *qualifiers* separated only by slashes.

dsd *qds* or *+qds* or *dsd+qds*
i.e. any number of *qds*'s separated by plus signs (but see below re concatenation).
+ may be replaced by an arbitrary number of +'s.

dsdlist *dsd* or *,dsdlist* or *dsd,dsdlist*
i.e. any number of *dsd*'s separated by any number of commas.

Examples:-

```
SYSIN   BL10.A
FROM = .A.B
INPUT = BL10.A+RJS1.X.Y
TO =  &TEMP
WITH = RJS1.PDS:MEM
WORK: = &A,&B,,&C
```

The last sets up WORK1 = &A
WORK2 = &B
WORK4 = &C

The *qualifiers* are used to specify the characteristics of a dataset, which are:

(a)	Size	Primary allocation, Secondary allocation, PDS directory blocks
(b)	Format	Record format(i.e. RECFM), Blocksize (i.e. BLKSIZE), Line width (i.e. LRECL)
(c)	Disposition	(i.e. status component of DISP)
(d)	Organisation	(i.e. DSORG) PO (or PDS) or PS
(e)	Miscellaneous	RLSE, OPT, etc.

Details of these *qualifiers* follow in section 4.3.3.

4.3.2 SPECIAL *NAMES*

The character % is used to distinguish special Phoenix *names*, which may be special *ddnames*, system commands or special *dsnames*. $ is used for dummy, while * introduces some datasets which are all terminal datasets in a console session and refer to various output datasets in a job.

SPECIAL *DDNAMES*

%COMMAND is the *ddname* required for command input at the next level.
(levels of command are described in 4.8.3 below).

%LIB is the *ddname* required for the step library at the current level.

COMMAND0 and COMMOUT0 are used for main input and output to Phoenix in jobs, see Section 2.3.2.

SYSABEND, SYSUDUMP, CAMDUMP, and CAMXDUMP are used for taking dumps, see Section 4.10.1.

STEPLIB and JOBLIB should not be used as *ddnames*.

Further *ddnames* starting with COMM or STEP are reserved for system use.

SPECIAL *DSNAMES*

System documents

%P	Previous document
%C	Current document
%R	Results document
%O	Output document
%M	Messages document

%A is an anonymous disc dataset (useful as work datasets)
Each occurrence of %A references a distinct dataset.
%H is used for a dataset provided instream (H means here) - see 4.5.8.
%N (Null) cancels the data definition. If it is the complete *dsd*, then the whole data definition is ignored. If it is part of a concatenation then that part is ignored.

%CL) (SYS2.COMMPROC
%SL) are abbreviations for (SYS2.COMMANDS
%XL) (SYS2.PHOENIX.C

%L is equivalent to %SL

Defaults for system documents

%R, %O, %M, %A default to MEDIUM space, VBL format
%H defaults to MEDIUM space, FB format

%H, %R, %O, %M all set blocksize to the largest possible multiple of logical record length less than 1000

Temporary dataset names starting with &COMM are reserved for system use.

NAMES OF DEVICES

Device names may have different meanings in a console session and a job, and are presented in tabular form. Terminal and dummy may be used for both input and output, while other devices are for output only.

name	Console session	Phoenix job
$	dummy	dummy
%DUMMY	dummy	dummy
*	terminal	shared SYSOUT=C
(Synonyms: %PRINT, %TERM, %TERMINAL)		
*$	terminal	dummy
**letter*	terminal	SYSOUT=*letter* see below
%OUT*digit* (Synonym: **digit*)	terminal	dataset whose *ddname* is COMMOUT*digit* - set up as *A if not preallocated
%PRINTA	terminal	SYSOUT=A) i.e. printer
%PRINTC	terminal	C)
%CARDS	terminal	B i.e. card punch
%PTAPE5	terminal	5)
7	terminal	7) i.e. paper tape punch
8	terminal	8)
%PLOT	dummy	SYSOUT=P plotter
%WIDEPLOT	dummy	Q wide plotter

Only those letters which are provided throughout the system are useful, i.e. A B C P Q R. Some of these facilities (C P Q R 5 7 8) are not implemented at the time of printing. Unimplemented and unrecognised letters are treated as *.

Other names starting with % are reserved.

All device datasets have default format unset (FN) and are always set up as unblocked. In a console session the terminal input defaults to F format with LRECL=terminal line width. (See TERMINAL WIDTH in Part II).

4.3.3 *QUALIFIERS*

Size *qualifiers*

	Primary (tracks)	Secondary (tracks)	Directory (blocks)
SMALL) S)	1	1	3
NORMAL) NL)	2	2	5
MEDIUM) MED)	2	5	5
LARGE) LA)	10	20	10
GIGANTIC) GIGA) G)	20	50	200
K*digit*) SP*digit*)	Preset space *digit*		(see SET SPACE in Part II)
TRKp,q,r	p	q	r)where p,q,r
CYLp,q,r	p	q	r)are *ints*

TRK sets up an allocation of p primary and q secondary tracks, with r directory blocks, while CYL allocates p and q cylinders. Items omitted are given the value zero; so that TRK5 has q and r replaced by zero. CYL allocations should only be used when really necessary.

Format *qualifiers*

	Record format	Line width (LRECL)	Blocksize
FA	FA	133	133
FB) FIXED)	FB	80	2480
FBA	FBA	133	1995
FBS) FIXEDS)	FBS	80	2480
FD	standard default		(see below)
FF	F	80	80
FN	null	0	0
FP	FA	121	121
U	U	0	4096
VAR) VARIABLE) VB)	VB	84	2498
VBA	VBA	137	2498
VBL	VB	136	2498
VBS	VBS	32756	2498
VBSL	VBS	136	2498
VBST	VBS	32756	13030
VBX	VBS	32768	2498 (=LRECL=X)

Fdigit Preset format *digit* (see SET FORM in Part II).

FD is the standard default, which is initially VBL but may be altered by a SET command.

FN leaves the format of a new dataset unset in which case the format must be supplied by the processing program. The format of an old dataset is left unchanged. FN as a qualifier of a here dataset causes the here dataset to take the format characteristics of the surrounding dataset.

The following may be used to set up explicit block sizes and line widths:-

B*int*) BLOCK*int*) sets up blocksize equal to *int*

L*int*) LRECL*int*) sets up line width equal to *int*

LX) LRECLX) sets up line width >32K

Disposition *qualifiers*

SHR	(synonyms	R, READ)
NEW	(synonym	N)
MOD	(synonyms	M, MODIFY, UP, UPDATE)
OLD	(synonym	O)

Organisational *qualifiers*

PO) PDS)	set for DSORG=PO. This is assumed if a member name is quoted.
PS	set for PS. This is the normal default: the only need for using /PS explicitly is to cancel a default setting of another organisation.
DA NODA	set for DSORG=DA

Miscellaneous *qualifiers*

null	the first character of the *qualifier* is a *qualifier-terminator*. The *qualifier* is ignored.
OPT	May be used as a *qualifier* when the *dsd* is a *ddname*. If the *ddname* does not exist the allocation is ignored for a SHR allocation, and is set up as *A otherwise.
NOTOPT	Cancel the OPT.
RLSE	Force RLSE; this is the default for PS datasets.
NORLSE	Force NORLSE; this is the default for PO datasets. Note that datasets are assumed to be PO if either a membername is quoted or if the PO (or PDS) *qualifier* is used. They are otherwise assumed to be PS.
VOL=*ser*	Allocate from volume whose *name* is *ser*. (Use of this *qualifier* is restricted).
CONTIG	Force contiguous allocation (only to be used if essential).
NOCONTIG	Normal allocation.
*	Comment *qualifier;* characters up to the next *qualifier-terminator* are ignored. In a default setting (i.e. %DEF=*qlist* or *qlist* free-standing) the remainder of the line is ignored.

At the start of a *command-environment*, standard default values for these characteristics are:

SIZE	2,2,5 tracks
FORMAT	VB 136 2498
DISP	SHR
ORG	PS unless membername given in the *dsname*
RLSE	if ORG is PS
NORLSE	if ORG is PO
NOTOPT	
NOCONTIG	

The default format of system documents and * devices is different; see 4.3.2 above.

Defaults may be altered as described in section 4.2.4 'Default settings' (above), and may be overridden by use of the *qualifiers* in a *qds*.

For example the default format may be altered to FB by:-

```
        %DEF = FB
or      /FB
```

If there are conflicting *qualifiers* in a *qds* then the left-most *qualifier* in the *qds* is used. Redundant or unnecessary information (e.g. directory blocks for a PS dataset) in a *qds* is ignored.

Examples of *qds*'s

```
.A/N/FB
PQRS.ABCD.FRED/PDS/TRK10,1,3
FROM
&WORK/N/VB/L100/B2125
```

4.4 EXAMPLES OF COMPLETE BASIC PHOENIX COMMANDS

```
:IEBCOPY(/N SYSUT3 = %A SYSUT4=%A /R/PDS       FROM      .SOURCPDS
           TO        .DESTPDS/OLD/NORLSE
            SYSIN = %H!)
 COPY   INDD=FROM,OUTDD=TO
!


:FILE (SYSPRINT=*,FROM=.TEXT(MEM),TO=*A)

BL10.LINKLIB:MYPROG ( %DEF  /R  FROM .INPUT)    (
      /N  TO    .OUTPUT/N/B800/L100
          WORK  &WORK/TRK1,5,5 %D='ANY TEXT' )
```

4.5 SEMANTICS OF DATA DEFINITIONS

The purpose of a data definition is to connect a *ddname* or group of *ddnames* with the requisite physical object or objects.

Basic phnysical objects are as follows.

4.5.1 DEVICES

TERMINAL
DUMMY
UNIT RECORD (printer, card punch etc.)

4.5.2 CATALOGUED DISC DATASETS

These have the form *name.name.name*, where the first *name* and the final *.name* are optional. These datasets are permanent and remain after the job or session that creates them has finished.

4.5.3 TEMPORARY DISC DATASETS

These have the form *&name*. Temporary datasets are deleted automatically at the end of the job or session that creates them, and also by use of the SCRATCH or DELETE commands.

4.5.4 WORKSPACE DISC DATASETS

These take the form %A, and are automatically deleted when the data definition is deleted.

4.5.5 *DDNAME* DATASETS

These specify a previously allocated object by quoting its *ddname*. The object being specified may be any previously setup *data-definition* (JCL or Phoenix). The *ddname* is renamed as the required *ddname* and renamed back when the allocation is deleted. It is therefore not possible to use one pre-existing *ddname* in this way more than once in a *command-environment*.

4.5.6 SYSTEM DOCUMENTS

The system documents referred to above (4.3.2) are temporary datasets that may be created automatically by the Phoenix system during the course of a job or an online session. The user refers to them by the Phoenix system *names* (%C %O etc.) instead of using an explicit temporary dataset *name* since the temporary dataset corresponding to a system document will vary during the course of a job or session, while the system *name* stays constant.

The user may refer to a system document explicitly, or implicitly via the default values in a *macro-command* (see 4.7).

For example:-

```
EDIT                    will edit %C to %O
TYPE %M                 will type the messages document.
```

There are five system documents:-

%P	Previous document
%C	Current document
%R	Results document
%O	Output document
%M	Messages document

Dataset characteristics may be set by *qualifiers* in the usual way; the default characteristics are as specified in 4.3.2; the defaults are used in the absence of explicit *qualifiers* or %DEF settings.

The standard commands provided by the Phoenix system are designed to arrange that the most recent active document is normally in %C; that %P contains a one level backup of %C; %M contains temporary output and %O or %R contain the next %C. In addition the contents of a newly created %R or %M (the typing documents) are typed at the console (or copied to the main spool offline) to give the effect of immediate results.

The way in which these documents are used will vary from command to command, but where they are used the following general rules apply:-

- %C and %P are always read only, and may not be set up other than SHR.
- %O, %R, %M may be read or written; OLD and MOD have the same effect as NEW.
- In any one command any one of %M, %R or %O may be set up; the only other combination that is allowed is %O with %M.
- At the end of execution of a command:-
  - If there is a new %O or %R (there cannot be both) that has been opened then %C is made to refer to this dataset and %P to refer to the old %C. The new %O or %R continue to exist and may be referred to explicitly.
  - If there is a new and opened %R or %M (there cannot be both) it is typed.
  - If %P was used as input dataset then the slaving of %C to %P will not take place.

Examples:-

EDIT	Edits %C to %O; at the end of the editing sequence the old %C will be found in %P, and the result of the editing will be found in %C (and %O).
EDIT TO %R	in addition to the above the new %C will be typed.
EDIT TO %M	The result of editing will be typed and available in %M, but %P and %C will not be altered.
EDIT %P	Edits %P to %O; at the end of the editing sequence the result of the editing will be found in %C (and %O); %P will be unaltered and the old %C lost.

4.5.7 MEMBERS OF DATASETS

Any of the disc datasets, described in 4.5.2 to 4.5.6, can be organised as Physical Sequential (PS) or Partitioned (PO). In the latter case it is frequently necessary to specify a member of the partitioned dataset that the command will access sequentially. This is done by appending

(name) or *:name* to the *dsname*.

If it is required to set up a PO dataset without specifying a membername then the *qualifier* PDS or PO must be used to tell the system that the organisation is PO.

4.5.8 INSTREAM (HERE) DATASETS

These datasets enable the user to supply a dataset inline, which is useful for small control datasets and in *command-sequences*, in which case a small dataset can be embedded in the sequence.

A here dataset is specified by

%H*here-delimiter*

The *here-delimiter* is a *name* or any single character other than) or space. The body of the here document is formed by the sequence of lines starting with the line that follows the one containing the closing bracket that ends the *command-environment*, and ending at a line containing only the *here-delimiter*, or at end-of-file if no such terminating line is found.

The instream dataset is formed into a dataset prior to the execution of the command. As described below (4.9.3) values for substitution of *parameter-occurrences* within the lines forming the dataset will take place during the process of forming the dataset.

If more than one instream dataset is defined in a *command-environment* then the text of the second dataset begins with the line following that which contains the first terminator, and so on for the subsequent datasets. Note especially that the several instream datasets must appear in the order in which they are specified in the *command-environment*, the leftmost one first.

Warning:

If the command specifying an instream dataset is not obeyed (for example after a syntax error, or a failing condition) then the instream dataset is not formed and the system will attempt to decode the lines of which it consists as if they were commands.

4.5.9 CONCATENATIONS

Up to 7 basic physical objects can be grouped together to form one composite physical object by using the operator +. For example, to concatenate .FILE1 and .FILE2 type

.FILE1+.FILE2

A concatenation may be continued on the next line by ending the first line with a +; the next line contains further *dsds* and may optionally start with a +.

Examples:-

```
CONCAT = .A+.B+
         .C+.D
CONCAT = .A+.B+
        +.C+.D
```

With certain limitations the program using a concatenation of *dsds* will treat it as one dataset. The limitations are:

- The concatenation must be for reading; that is, it must have disposition=SHR.

- Unless the processing program has been specially written to cope with concatenations properly then -

  The RECFM (F V or U) and logical record length of all members of the concatenation must be the same; the blocksize of any subsequent member must not be larger than that of the first member.

- Nothing may be concatenated after DUMMY or TERMINAL.

- The program must be using one of the standard access methods QSAM, BSAM or BPAM.

- A member of a concatenation may be specified as a *ddname* provided that the *ddname* is not itself a concatenation of datasets, and that it refers to a disc dataset.

- JCL ddnames may not be concatenated using the Phoenix language.

Most of the standard Phoenix commands (FILE, EDIT etc.) have been written to deal properly with concatenations.

4.5.10 *DDname lists*

Several *dsd*'s may be allocated in one *environment-element* by use of

ddseq = dsdlist

The *ddseq* is a *ddname* with one of the characters (other than the first) replaced by a :, and the *dsdlist* is a set of *dsd*'s separated by commas. These *dsd*'s can refer to any dataset.

Example:

```
SYSUT:0 = ,%A,$,,&FRED,*B
```

which sets
```
SYSUT20=%A
SYSUT30=$
SYSUT50=&FRED
SYSUT60=*B
```

There may be up to 9 *dsd*'s in a *dsdlist*.

4.6 PHYSICAL CHARACTERISTICS OF DATASETS

When specifying a *dsd* there is a lot of information that may need to be supplied in addition to the *ddname* and the physical specification. This extra information may be supplied by use of the *qualifiers*.

4.6.1 SIZE

The size of a dataset may be given by the *qualifiers* explicitly in terms of tracks or cylinders for primary, secondary and directory allocations or implicitly by means of the standard abbreviations.

For example:-

/TRK2,3,2 .or /MEDIUM

4.6.2 FORMAT CHARACTERISTICS

These are

RECFM (record format)
BLKSIZE (block size)
LRECL (line width)

As specified above (in 4.3.3) Phoenix provides a number of standard words describing common formats (VB for VB/L84/B2498 for example), as well as providing means for specifying any required format.

For example, to specify
RECFM=FB,BLKSIZE=800,LRECL=100
the *qualifiers* to use are
/FB/B800/L100

Explicitly provided B and L values override the B and L values associated with the standard format *qualifiers*. Hence

.T/FB/B800/L100
/B800/L100 .T/FB
/FB .T/B800/L100

are all equivalent.

4.6.3 STATUS

This is specified by the *qualifiers* described above, which are the standard IBM set (SHR, OLD, NEW, MOD) with some synonyms. The Phoenix system provides the following features that are additional to the IBM system:-

- A disc dataset that is specified as NEW and already exists will be treated as if OLD had been specified; that is, the old copy is overwritten; in particular the space allocation and DSORG may not be changed. The format of the old dataset is left unchanged unless the dataset is sequential and a new format is explicitly supplied. There is however nothing to prevent a program from arbitrarily altering the format of a dataset.
- A disc dataset specified as SHR or OLD that does not already exist will cause an error.
- The default status is SHR except for devices (* etc.) and %A datasets which have an assumed status of NEW.

4.6.4 ORGANISATION

This may be-one of

PS	physical sequential
PO)	partitioned
PDS)	

The Phoenix system assumes that a *dsname* which includes a member name refers to a member of a pds, and that other datasets are PS. The *qualifiers* PS, PO, PDS are provided to override this assumption. The only common case where these *qualifiers* are needed is where a new dataset is being allocated as a pds.

For example :-

```
SETUP (A,X/N/PDS)
```

The main use for the *qualifier* PS is to cancel a previous default PDS *qualifier*. The PS *qualifier* cannot normally be used to override the organisation of an existing pds since many Phoenix commands will not allow a pds to be destroyed in this way.

4.6.5 OTHER CHARACTERISTICS

RLSE, CONTIG and OPT are described above in 4.3.3.

4.7 *MACRO-COMMANDS*

Most standard commands are provided with a simplified calling sequence, known as a MACRO, which enables them to be used without the user needing to know all of the standard details of the *lsd*'s and other data needed by the command. This permits these commands to be used with a minimum of typing. A command being used via its macro calling sequence is known as a *macro-command*.

The system holds a *macro-definition* for each available macro which contains a list of the requisite *keywords* and associated data items.

The syntax for calling a *macro-command* is :-

macro-name [*keyword-value-pair*] [*keyword-value-pair*] .....

where the *macro-name* is a *name*(or *%name* for system macros) and a *keyword-value-pair* is

[keyword [=]] value

The *keyword* is a *word* and the nature of the value depends on the particular *macro-name* and the *keyword*.

If a *keyword-value-pair* is followed by a space and : then further *keyword-value-pairs* are taken from the next line.

A *keyword* is recognised by the system as being a *word* ending in a space or =. The user may supply a value without its *keyword;* this will be accepted as the value for the first unused *keyword* provided that:-

(a) No previous *keyword* has been supplied.
(b) The specification of the macro does not insist that the key be quoted.

A *word* which does not match any key for the macro will be treated as an unkeyed value. *Keyword-value-pairs* for which no data is supplied are given default values, unless the specification of the macro insists that an explicit value is given, in which case the system indicates an error.

There are four types of key

(a) The *keyword* may be omitted and the data need not be supplied.
(b) The *keyword* may be omitted but data must be supplied.
(c) Data need not be supplied but if it is, then the *keyword* may not be omitted.
(d) Both the *keyword* and the data must be supplied.

The information in the macro, and the values supplied by the user, are used to construct a command line which may be either a basic command or a macro.

Users are referred to Part II for the description of each of the standard set of *macro-commands* and the use of the various types of *keyword-value-pairs*, and to section 4.11.5 and Appendix E for the use and creation of user-constructed macros.

If the *macro-name* quoted is not in the standard set nor in any set loaded by the LIBRARY command, then the command may be a procedure (4.8.2). If the *name* is not in any available procedure library then the system indicates an error.

4.8 *COMMAND-SEQUENCES*

Phoenix provides a means whereby commands can be obeyed from a *command-sequence* contained in a dataset.

4.8.1 C-COMMANDS

The C-command is used to call a *command-sequence*. The syntax is a particular case of a macro calling sequence, and may be used in one of two alternative forms:-

(a) C [*dsd*] [$\begin{pmatrix} P \\ PARS \end{pmatrix}$ *parameter-declaration*]

(b) C [*dsd*] (*command-environment*) [*parameter-declaration*]

As usual, a *keyword* may be followed by an optional =.

In the first form the *keyword* P or PARS need only occur if the *dsd* is omitted or if the first word in the *parameter-declaration* is P or PARS.

For botn forms the *parameter-declaration* is used to set parameters as described below (section 4.9). If the *dsd* is omitted it defaults to %C. The *command-sequence* specified is the contents of the *dsd*.

4.8.2 PROCEDURES

The syntax of a call of a procedure is the same as that of a *command-sequence* except that the C *dsd* is replaced by *name*, where *name* is the name of the procedure. Thus, we have

name [$\begin{pmatrix} P \\ PARS \end{pmatrix}$ *parameter-declaration*]

and

name (*command-environment*) [*parameter-declaration*]

A procedure call is actually a call to a *command-sequence* kept on a dataset particular to the procedure, and all the facilities of a C command apply to a procedure.

The system holds a list of available procedures in the same way as for macros; extra sets of procedures may be loaded by the LIBRARY command. If the *name* is neither in any available macro library nor in any available procedure library then the system indicates an error. Note that the search for a system macro *name* takes place after that for a procedure *name* but that *%name* in name position will always cause the system to retrieve the system macro with that *name*.

4.8.3 LEVELS AND NESTING

Obeying the C command causes the *dsd* to be used as command input. This is referred to as 'taking commands from the next level'; the level containing the C command is referred to as the 'current level' or 'calling level'.

Any *command-sequence* may contain a C command, and by this means *command-sequences* may be nested. The maximum nesting depth is 9, but because of limitations on buffers and dataset allocations it may not always be possible to reach the maximum depth.

Parameters (see 4.9 below) set in the C command are local at each level; *parameters* of this level may be passed to the next level by starting the *parameter-declaration* with a +. The *data-definitions* set up in the *command-environment* of a C command remain set up for all enclosed levels. %S and %T settings act as defaults for enclosed levels.

%LEVEL (see 4.9.5) is set to the level number of the current *command-sequence*.

Taking a *command-sequence* from the terminal, i.e.

```
C *
```

online requires interactive typing of commands, and the user will be prompted by

```
READY -n
```

where n is the level number. The initial command level is level 1, and READY is used as an abbreviation for READY -1. In some circumstances READY. is used instead of READY -n (see Section 4.11.1).

The *dsd* in the C command may be an instream dataset. In this case the *command-sequence* consists of the body of the instream dataset, with substitution for *parameter-occurrences* taking place as appropriate to the level containing the C command (see 4.9.3).

Examples:-

(a) After C .A at level 1
the *command-sequence* in .A is being obeyed at level 2.

(b) After C %H! at level 1
TYPE .A
.....
!
The instream dataset forming the *command-sequence* is created at level 1, and the commands are obeyed at level 2.

4.9 *PARAMETERS*

4.9.1 *PARAMETER-NAMES*

Parameters are provided as a means of passing data to the *command-sequence* being called. They may be tested in a *conditional-command* (see 4.10.5). *Parameters* have a name and may have a value. If they have a value, they are said to be set. If not they are unset. Note that they may have the value of the null string and will then be set. *Parameter-names* may be any of:

Any single letter or digit	0, 1, ..9, A, B, ...Z
Any multicharacter item consisting of up to 8 letters or digits	A1, 8BCD, TWENTY1

Witnin *parameter-names* lower case is equivalent to upper case. Up to 14 multicharacter and any of the single character *parameters* may be specified in the *parameter-declaration*. *Parameters* specified in this manner within the C command are local to the next level. The local *parameters* appropriate to the current level are restored when control is returned to the current level. In addition global *parameters* may be set by the PARS (or SET P) command (see below 4.11.2 and Part II).

4.9.2 *PARAMETER* VALUES

The value of a *parameter* is a string (but see below for *special-parameters* %Ln and %Gn) and is specified within the *parameter-declaration*. This is defined as an optional + followed by any sequence of the forms described below separated by one or more spaces. A *parameter-declaration* may be up to 255 characters in all.

(a) *parameter-name* [=] *text*
quoted-text

or

(b) <*text*> where *text* may not contain >

or

(c) - *parameter-name*

If + is present at the beginning of the *parameter-declaration* then any local *parameters* set at the current level and not reset in the *parameter-declaration* become set local *parameters* for the next level.

Form (a) sets the corresponding *parameter* with the given value.

Examples:
```
A= FRED
B23 = 'JACK''S PEN'
8C  JACK'S
```

In the third example the optional = has been omitted.

Form (b) may be used to set *parameters* 1 to 9. The first occurrence sets *parameter* 1, the next 2, etc. The *text* within <> may contain any character other than >. If the first character after the < is > then the corresponding *parameter* is left unset.

Example:
```
<ABC><><DEF>

      sets   1  to ABC
             3  to  DEF
             and leaves 2 unset.
```

The two forms can be mixed; for example-

```
<ABC> A=FRED <>  B23='JACK''S PEN' 8C JACK'S <DEF>
```

combines the two above examples.

Form (c) is used to unset *parameters* and is useful to negate part of the effect of a + in the *parameter-declaration*.

Examples of *command-sequence* calls:-

```
C  PQRS.ABC
C  PQRS.ABC  A1=JACK
C  PQRS.ABC + A1=JACK -FRED
C  PQRS.ABC (PROGRAM=.A)
C  P A1=JACK  (commands taken from %C)
C  PQRS.ABC P P=252 A1=JACK
```

and of procedure calls:-

```
FTG1CLG (PROGRAM=.INPUT) FT05=.A SECG=120
MLI FROM .SOURCE TO .OUTPUT
```

4.9.3 *PARAMETER* SUBSTITUTION

The values of *parameters* are substituted into *parameter-occurrences* which are

<*parameter-name*>

or

<*parameter-name*$*textl*>

or

<*parameter-name*$*parameter-occurrence*

where *textl* is *text* not containing >

Thus, in words, a *parameter-occurrence* can take three forms

(a) *parameter-name* enclosed in < and >
(b) *parameter-name* followed by $*text*, all enclosed in <>
(c) *parameter-name* enclosed in < and $ followed by another occurrence.

Any number of *parameter-occurrences* may occur anywhere within a *command-sequence*. All three forms specify an occurrence of the *parameter* whose name is *parameter-name;* they differ in the specification of the default that is used if the *parameter* is not set.

Form (a) - no default is provided.
Form (b) - the *textl* is the default.
Form (c) - a further *parameter-occurrence* is the default.

Substitution rules:-

- If the switch NOSUB (see 4.10.2) is set then no substitution takes place (Note that NOSUB is initialised as set in level 1)
- If the switch NOSUBH (see 4.10.2) is set then no substitution takes place within here datasets at that level.
- If the *parameter* has a set value, then that value replaces the *parameter-occurrence* in the text.
- If the *parameter* is unset, then the default is used, and if no default is provided then the *parameter-occurrence* is omitted.
- Substitution takes place from left to right; the text after substitution will not be scanned for *parameter-occurrences*.

Examples:-

If the following is obeyed within a *command-sequence*, with

	A	set to the *text*	AAA
and	B	set to the *text*	<1>
and	C	unset	

then

1<A>2 is replaced by	1AAA2
1<B>2	1<1>2
1<C>2	12
1<C$CCC>2	1CCC2
1<A$XXX>2	1AAA2
1<C$<A>2	1AAA2

4.9.4 LOCAL AND GLOBAL SETTINGS

A *parameter* is set and has a value if either

(a) it is set locally
or (b) it is set globally (by the PARS, REQUEST or SET commands)
or (c) it is preset (U, S, etc: see 4.9.5 and 4.11.4)

If more than one apply then (a) overrides (b) which overrides (c).

The substitution of *parameter-values* for *parameter-occurrences* in the *command-sequence* being called takes place line by line as each line is decoded. Thus a *parameter* set globally within a *command-sequence* will affect substitutions in following lines of the *command-sequence*.

A here document is formed before the command that uses it is obeyed, so that,in particular, a here document being used as a *command-sequence* has the *parameters* of the calling level substituted into *parameter-occurrences* within it unless NOSUB is set at the calling level.

Example:

```
SW -NOSUB
PARS A=1
C %H!   A=2
PARS A=3
*B<A>
!
```

the value of the label is B1.

4.9.5 *SPECIAL-PARAMETERS*

Special-parameters all start with a %. They may be used in the same way as normal *parameters* for substitution in commands and may be tested by the IF command. The values of some of them may be set by the user and the remainder are set by the system. Some contain textual values in the same way as normal *parameters*, and some contain only numeric values, and the remainder may contain either. One set of *special-parameters* (%L and %G) are entirely set and used by the user, while the remainder are used for communication between the user and the system. When numeric *special-parameters* occur in a *parameter-occurrence* the substituting text is a character representation of the integer value.

SETTING SPECIAL PARAMETERS

Special-parameters always have a set value, either by initialisation, or by explicit user setting or by implicit system setting.

Users may set *text* values in *special-parameters* by

(a) SET / PARS *special-parameter quoted-text*
or by using the REQUEST command (see Part II specifications)

and general values by

(b) SET / PARS *special-parameter var-string*
or by using the REQUEST command (see Part II specifications)

where *var-string* consists of *special-parameters* and *ints* separated by + or -. If the first item of the *var-string* is a *special-parameter*, then the *special-parameter* being set is set to the same type (textual or integer).

When setting a text value with either form (a) or (b) the right hand side of the assignment may be followed by an optional

,int

in which case the value is truncated to *int* number of characters.

Examples:

'ABC',1 is the character A

and if %L2 contains ABCD

%L2+1,2 is the string BC

as the +1 moves on the address pointer one byte for *text* values.

User *Special-parameters*

These take the form:

%L*digit* (L = Local)

%G*digit* (G = Global)

where *digit* = 0, 1, 2, 3, 4, 5

These may be set freely by the user to any *textual* or *numeric* values.

They are initialised at the start of the session or job to numeric zero.

The %L's are local to a command level; on entry to a level they are initialised to the value they currently hold in the calling level, on return to the calling level their values are restored to the values they had on entry.

The %G's are global and retain their values until explicitly reset or cleared.

The CLEAR command resets all %G's and %L's to zero.

System *Special-parameters*

These take the form:

%QLEVEL is user settable and contains an integer value which sets the *command-level* beyond which command reflection and warning message production does not take place. It is initialised to 15.

%RCLEVEL is user settable and contains an integer value which is the value of a return code at or below which abnormal sequencing will not take place (see 4.10.7). It is initialised to zero.

%LEVEL is set by the system to an integer which is the current *command-level*.

%SCC is set by the system to the system completion code of the last abending command (this is an integer although OS system abend codes are usually quoted in hexadecimal).

%UCC is an integer which is set by the system to the user completion code of the last abending command.

%RC is an integer which is set by the system to the return code of the last command.

%ABEND is set by the system to the abend code of the last abending command in system format ie. %ABEND = %SCC*2**12 + %UCC.

%USER has a textual value which is set by the system to the userid given in the job description or to the userid of the logged-on user.

%SET has a textual value which is set by the system to the SET name (4.11.2)

4.10 CONTROL FEATURES

The following features are provided to organise the flow of control in *command-sequences*.

4.10.1 ABENDS

A command which ends in an ABEND causes a suitable message; a dump of the ABEND can be obtained by preallocating (in the environment or by the SETUP command) ddname(s) SYSABEND, SYSUDUMP, CAMDUMP or CAMXDUMP as required, and setting the switch DUMP.

A command that ends with non-zero return code greater than %RCLEVEL will cause a suitable message if the switch ERRCODE is set.

4.10.2 FLAGS AND SWITCHES

Flags are set by the system and may be tested by the user; switches may be set and tested by the user.

ERROR FLAGS

All errors set internal flags that may be tested by command, and which in normal circumstances interrupt the flow of control; see 4.10.7. The possible errors and the associated *flag-names* are:-

flag-name	error
ABEND	command ended in ABEND
RC)	command ended with non-zero return code
CC)	
BADFILE	an error occurred while trying to create a data definition
BADSYNT	a command line contained a syntax error
BADPARM	error in the *parameter-declaration* for the C command
BADP	error in a *parameter-occurrence*
ERROR	any of the above
OLDABEND	some previous command in the current level ended in ABEND

OTHER FLAGS

In addition to the flags mentioned above there are the following which are not connected with error states.

flag-name	
ONLINE	True during a console session
TRUE	Always true
FALSE	Always false
FAST)	True in a Phoenix Monitor job
BATCH)	

SWITCHES

Switches, unlike flags, can be set or reset as well as tested. They are used to control certain options described below, and are fully described in Part II.

switch-name	option in force when switch is set
COMMENT	allow reflection of comments starting ** even if QUIET set.
DUMP	cause dump after abend if suitable *ddname*(s) provided
ERRCODE	output return codes

HOLDSP	hold space for abend processing
NOEXEC	non-control commands are interpreted but not obeyed
NOSTOP	modify error handling - see 4.10.7
* NOSUB	substitution of parameters is suppressed
* NOSUBH	substitution of parameters in here datasets is suppressed
QUIET	suppress command reflection
REFLECT	allow reflection of expansions of commands
* RESTORE	user settable switches reset on return to previous level
SPOOL	output copied to a user supplied dataset

* These switches apply at the current level only and are initialised to the off state at each level except that NOSUB is set at level 1.

There are also 16 numbered switches which are available to the user referred to as SW0, SW1, ..., SW15.

4.10.3 CONTROL COMMANDS

The following control commands are built-in to the Phoenix system. They are described more fully in the referenced sections.

Comments and labels - see 4.10.4

blank line	is ignored
**name rest*	*name* is a label, *rest* is comment
** rest*	comment line
***rest*	comment line that is reflected if COMMENT is set

Jump commands - see 4.10.4

JUMP
SKIP
RETURN
SET ERRJUMP
SET RETURN

Conditional commands - see 4.10.5

IF
UNLESS

Termination commands - see 4.10.6

%FINISH	the built-in command accessed by the macro FINISH
LOGOFF	synonym for %FINISH
QUIT,Q	
ABANDON,A	
YES	
NO	

Dataset control commands

| **%%SPOOL** the built-in command accessed by the macro SPOOL
| SCRATCH
| SETUP

Set internal data commands - see 4.11.2

SET
CLEAR
PARS
SWITCH,SW

Miscellaneous

| **%%C** the built-in command accessed by the C macro
| LIBRARY - see 4.11.4
NULL see 4.10.6
WAIT see 4.10.6

4.10.4 LABELS AND JUMPS

A command line starting

**name*

defines the occurrence of a *label-name* which is a *name;* the remainder of the line is taken as comment.

The command

JUMP *label-name*

transfers control to the first occurrence of the *label-name* in the current *command-sequence* provided that it is rewindable

while the command

SKIP *label-name*

transfers control to the first occurrence of the *label-name* subsequent to the SKIP command in the current *command-sequence.*

If the *label-name* is omitted in either JUMP or SKIP commands the command is ignored.

If the *label* being jumped to cannot be found in the current *command-sequence*, the effect is the same as QUIT.

The command

RETURN

is equivalent to

JUMP *return-label*

where *return-label* has been set by the command

SET RETURN *return-label*

The RETURN feature is useful for a part of a *command-sequence* that is used as a subroutine. For example:-

```
*YY
....
RETURN
```

could be called by

```
SET RETURN XX
JUMP YY
*XX
```

4.10.5 CONDITIONAL COMMANDS

A command may be obeyed conditionally. This is specified by prefixing the command by one or more *conditional-clauses*; if a *condition* has the value TRUE the remainder of the line is obeyed. Otherwise the next command in the current *command-sequence* is obeyed. The word ,O may be inserted after the final *conditional-clause*.

A *conditional-clause* is defined as:-

IF / UNLESS *condition*

and a *condition* is an optional + / - / NOT followed by

switch-name
flag-name
special-parameter relation int
special-parameter relation quoted-text
parameter-name IS SET
parameter-name IS UNSET
parameter-name relation text
parameter-name relation quoted-text

Several *conditions* may be connected by the logical operators AND and OR to form one compound *condition* which is evaluated from left to right.

The precise composition rules are:

the value of A R B

where A and B are *conditions* and R is one of AND or OR is given by the following rules:

- if A is true and R is AND, then the value is that of B.
- if A is true and R is OR then the value is TRUE.
- if A is false and R is AND the value is FALSE.
- if A is false and R is OR, then the value is that of B.

Conditions are evaluated from the left in accordance with the above rules, IF between conditions having the same effect as AND, until a condition is found that ends in anything other than IF, UNLESS, AND or OR.

UNLESS is the same as IF NOT.

Relations are:-

EQ	equal
NE	not equal
H	higher
GT	greater
GE	greater or equal
NL	not less
NH	not higher
LE	less or equal
L	less
LT	less than

Comparisons are arithmetic for *special-parameters* of integer type and by the EBCDIC collating sequence otherwise. A string has a lower value than any longer string that starts with the same characters.

Examples:-

```
IF ONLINE TYPE .X
IF %GO EQ 20 JUMP FRED
IF PAR20 IS SET IF PAR21 EQ THIRTYFIVE DO C .A
IF QUIET AND NOT ONLINE OR SW1 QUIT
```

4.10.6 OTHER CONTROL COMMANDS

QUIT
Q [n]

QUIT returns control to the calling level. The return code is set to n if present, otherwise it is left unaltered. In a job, QUIT at the initial level is the same as FINISH, and also passes a return code. In a console session QUIT at the initial level is the same as NULL.

ABANDON
A [n]

ABANDON returns control to the nearest interactive level in a console session (i.e. to a READY) and causes step end in a job. In the normal case of a one step job this will cause end of job. The return code is set to n if n is present.

FINISH)
LOGOFF) These commands terminate a console session and cause
%FINISH) step end in a job.

YES is equivalent to QUIT 0
NO is equivalent to QUIT 16
WAIT [n] waits for n secs (default 10 secs); ignored in a job
NULL [n] No action other than setting return code to n if present

There is an implicit QUIT when end-of-file is encountered in a *command-sequence*.

4.10.7 FLOW OF CONTROL AFTER ERROR

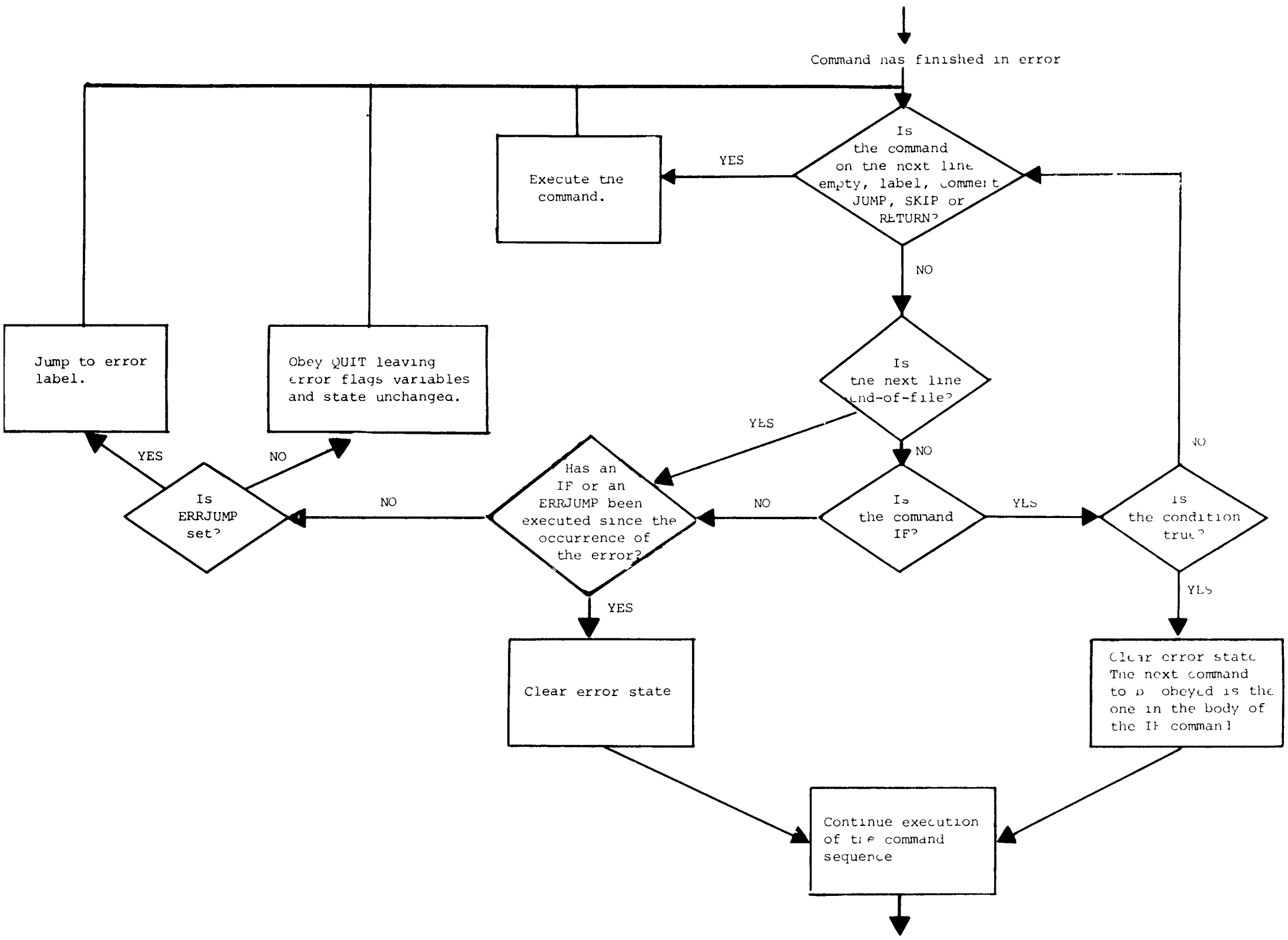
Abnormal sequencing as described below takes place after a command line which causes an error flag to be set. To avoid this abnormal sequencing the switch NOSTOP may be set. In this case the ERROR flags are still set, and the associated values may be tested but no implicit change of sequencing takes place.

The command

SET ERRJUMP *error-label*

may be used to set an *error-label* which is a *name* and is used as a *label* in an implied JUMP command in error circumstances, as described below.

After an error the lines following the command are processed as follows:-


Command has finished in error
Is the command on the next line empty, label, comment JUMP, SKIP or RETURN?
YES
Execute the command.
NO
Is the next line end-of-file?
YES
NO
Is the command IF?
NO
YES
Is the condition true?
YES
NO
Has an IF or an ERRJUMP been executed since the occurrence of the error?
NO
YES
Is ERRJUMP set?
YES
NO
Jump to error label.
Obey QUIT leaving error flags variables and state unchanged.
Clear error state
Clear error state The next command to be obeyed is the one in the body of the IF command
Continue execution of the command sequence

Note that the UNLESS command is treated exactly the same as IF NOT.

This has the effect that errors are dealt with either locally by IF statements or globally with an ERRJUMP. If neither are provided a QUIT is obeyed. All the above error handling is suppressed if the switch NOSTOP is set, in which case commands are obeyed regardless of previous errors.

The command

NULL

may be used to force examination of the error state, for example in the case that the next command is an IF which is not to be obeyed if the command ends in an error state.

Example:-

```
EDIT .A
NULL
IF SWO FILE .B
```

4.11 MISCELLANEOUS

4.11.1 INTERRUPTING COMMANDS

The BREAK key can be used to interrupt a command, or output, in a console session. The immediate response to BREAK is ! and further responses depend on the individual command being obeyed. The standard Phoenix response is:-

```
***-BREAK
```

after which the user should type one of the following characters and a newline, which causes the effect described:-

A	causes an ABANDON
Q	causes a QUIT
N	(Null) - break is ignored
F	Fault - abend F22 is sent to the program interrupted
newline	same as N
other	reserved- treated as N.

Whenever BREAK is used the current input and output buffers are lost, so that BREAK N during output will step the output on a little. If the terminal system is accepting input then @I may be used instead of BREAK.

STANDARD BREAK SEQUENCES

After a READY and before a command has been completely input, BREAK will cause a response of

READY.

Some commands (eg. EDIT, PATCH) have their own break processing; for example EDIT puts out ***I and returns to its command input.

REPEATED BREAKS

Using BREAK for a second time before the first BREAK has been processed causes the break processor to return one level. For example breaking EDIT's ***I causes entry to the standard Phoenix break sequence, and hence a ***-BREAK to appear. An N at this point returns to EDIT.

Breaking the standard sequence (BREAK before the system types ***-BREAK or instead of the control character) causes entry to the break sequence of the Phoenix supervisor. This causes re-entry to the Phoenix system and a message

***SESSION INTERRUPTION

after which the user should type A, F, Q or N as described earlier in this section. All data definitions and local *parameters* are lost, as well as any private libraries. In the latter case the system puts out the information message

PRIVATE LIBRARY CANCELLED

However, all global *parameters*, temporary datasets, and system documents are preserved.

4.11.2 SET COMMANDS

This collection of commands is provided to set data which will be used by the Phoenix command system. The SET command (see Part II) has the following subcommands:-

SET [NAME] *name*		sets up *name* as the 'set user'. NAME may be omitted provided that *name* does not clash with any of the other keywords.
[SET] SWITCH *switch-name*)		set switches
[SET] SW *switch-name*)		
[SET] CLEAR		unset all global *parameters*
[SET] PARS *parameter-declaration*)		global parameters (as 4.9,
SET P *parameter-declaration*)		but without an optional +).
SET *special-parameter var-string*		value of *special-parameter*. *Var-string* is any sequence of *integers* or *special-parameters* separated by + or -.
SET ERRJUMP *label*		set value of *error-label*
SET RETURN *label*		set value of *return-label*
SET CHAR *word*	character	set special character (4.11.3)

SET	SPACE*digit*)		
SET	SP*digit*)	data	set space data
SET	K*digit*)		
SET	FORM*digit*)		
SET	F*digit*)	data	set format data
SET	TERMINAL *word integer*		

These are described more fully in Part II. Those commands whose data is not the rest of the line may be followed by other set items on the same line, with the word SET omitted.

Example:-

```
SET NAME SPQR SW QUIET
```

4.11.3 SET CHAR COMMAND

This command is used to alter locally the identity of standard Phoenix control characters at the current level. The commands available are:-

a) SET CHAR BRA character

the character is used instead of the character <

b) SET CHAR KET character

the character is used instead of the character >

c) SET CHAR QUOTE character

the character is used instead of the character '

d) SET CHAR END character

a line ending with the specified character is concatenated to the next line (with the continuation character removed) subject to the maximum length of such a concatenation of lines being 255 characters.

4.11.4 INITIAL STATE

(a) Console session

After LOGON the user is obeying commands interactively at level 1. Global Parameters U and S are set to the userid, and all others are unset.

Terminal width is set to 72 for teletypes or 80 for some vdu's on fixed lines, and 120 for 2741's.

Switches NOSUB and COMMENT are set; all others are unset.

Of the *special-parameters* %QLEVEL is set to 1, %USER and %SET are set to the userid; others are set to zero.

The LOGON process obeys any command sequence set up by the user with a MKINIT command (see Part II) and then the standard system initialisation which is kept in %XL:INIT.

(b) In a job

The commands on the dataset COMMAND0 are obeyed at level 1; system output is sent to COMMOUT0. Both these datasets are set up automatically by the JCL procedure that invokes Phoenix.

Parameters and switches are set as above with the addition that ERRCODE is set.

The standard procedure library (called NORMAL) is available. (See LIBRARY in Part II).

4.11.5 PROVISION OF EXTRA MACRO LIBRARIES

Extra system or private macro libraries or procedure libraries may be made available using the LIBRARY command.

A private library is loaded by the command

LIBRARY *dsname*

where the *dsname* describes a loadable member of a pds. A system library is loaded by the command

LIBRARY *name*

where the *name* is one of the set of available libraries (see Part II).

Note that it is not possible to load two libraries with the same *name* from two different pds's, so that

```
LIBRARY   .A:MEM
LIBRARY   .B:MEM
```

will load only the first one.

The most recently loaded library may be deleted by

LIBRARY CANCEL

and all libraries by

LIBRARY CANCEL ALL

A private procedure library may be constructed by the command MKLIB.

For information on the construction of private macro libraries see Appendix E.

APPENDIX A

CHARACTER CODES TABLE

The following table gives the definitive graphic assignments to EBCDIC (the internal code used on the 370) and also the codes and graphic assignments for ASCII and 7 track Titan Flexowriter paper tape codes. The EBCDIC column is in hexadecimal and the Flexowriter columns are in octal with the sprocket hole between the 014 (octal) bits. The same ASCII assignments apply to input from ASCII devices other than the paper tape readers. Blanks mean no assignments. The parity bit for the ASCII codes is not included but if needed it is the left hand bit and the recommended parity is even The parity bit for the Flexowriter codes is 020 (octal). Further details about paper tape I/O may be found in Chapter 3 of this manual.

The following Flexowriter codes have no EBCDIC equivalent

Shift lower 26 (octal) (in lower and upper cases)
Shift upper 7 (octal)

The punched card code and the decimal and binary representation of the hexadecimal EBCDIC code may be found on the yellow card 'System/370 Reference Summary' obtainable from the Computing Service Bookshop. Note that the graphics defined in the yellow card for EBCDIC differ from those below

Graphics marked with + are not available on the printer. To input characters that have no code equivalents escaped characters as defined later may be used.

EBCDIC Hex	EBCDIC Decimal	GRAPHIC	ASCII Octal	ASCII Hex	FLEXOWRITER UC	FLEXOWRITER LC	
00	0	NUL+	000	00			Null
01	1						
02	2						
03	3						
04	4						
05	5	HT+	011	09	004	004	Horizontal Tab
06	6						
07	7	DEL+	177	7F			Delete
08	8	GE+					Graphic escape
09	9						
0A	10						
0B	11	VT+	013	0B			Vertical Tab
0C	12	FF+	014	0C			Form Feed
0D	13	CR+	015	0D			Carriage Return
0E	14	SO+	016	0E			Shift Out
0F	15	SI+	017	0F			Shift In
10	16						
11	17						

EBCDIC		GRAPHIC	ASCII		FLEXOWRITER		
Hex	Decimal		Octal	Hex	UC	LC	
12	18						
13	19						
14	20						
15	21	NL+			002	002	New Line
16	22	BS+	010	08	025	025	Backspace
17	23						
18	24						
19	25						
1A	26						
1B	27						
1C	28						
1D	29	IGS+	035	1D			
1E	30						
1F	31	IUS+	037	1F			
20	32						
21	33						
22	34						
23	35						
24	36						
25	37	LF+	012	0A			Line Feed
26	38						
27	39	ESC+	033	1B			Escape
28	40						
29	41						
2A	42						
2B	43						
2C	44						
2D	45						
2E	46						
2F	47	BEL+	207	87			Bell
30	48						
31	49						
32	50						
33	51						
34	52						
35	53						
36	54						
37	55						
38	56						
39	57						
3A	58						
3B	59						
3C	60	DC4+	024	14			Stop code on paper tape
3D	61						
3E	62						
3F	63						
40	64	Space	040	20	020	020	
41	65						
42	66	[	133	5B		061	
43	67	]	135	5D		062	

EBCDIC Hex	EBCDIC Decimal	GRAPHIC	ASCII Octal	ASCII Hex	FLEXOWRITER UC	FLEXOWRITER LC	
44	68	{+	173	7B			
45	69	}+	175	7D			
46	70						
47	71						
48	72						
49	73						
4A	74	£					
4B	75	.	056	2E	057		
4C	76	<	074	3C		043	
4D	77	(	050	28		070	
4E	78	+	053	2B	075		
4F	79	|	174	7C		067	
50	80	&	046	26			
51	81						
52	82						
53	83						
54	84						
55	85						
56	86						
57	87						
58	88						
59	89						
5A	90	!	041	21			
5B	91	$	044	24	054		§ in Flexowriter code
5C	92	*	052	2A		076	
5D	93	)	051	29		051	
5E	94	;	073	3B		073	
5F	95	¬	176	7E	073		Alternative graphic ~
60	96	-	055	2D	076		
61	97	/	057	2F	037		
62	98	\	134	5C			
63	99	V			052		
64	100	Λ				052	
65	101	↑	136	5E			
66	102	←	137	5F		054	→ in Flexowriter code
67	103						
68	104						
69	105						
6A	106	^+	136	5E			
6B	107	,	054	2C		057	
6C	108	%	045	25			
6D	109	_	137	5F		046	
6E	110	>	076	3E		064	
6F	111	?	077	3F			
70	112						
71	113						
72	114						
73	115						
74	116						
75	117						

EBCDIC Hex	EBCDIC Decimal	GRAPHIC	ASCII Octal	ASCII Hex	FLEXOWRITER UC	FLEXOWRITER LC	
76	118						
77	119						
78	120						
79	121	` +	140	60			
7A	122	:	072	3A		037	
7B	123	#	043	23			£ on some ASCII devices
7C	124	@	100	40		075	χ in Flexowriter code
7D	125	'	047	27		040	
7E	126	=	075	3D		045	
7F	127	"	042	22			
80	128						
81	129	a	141	61		121	
82	130	b	142	62		122	
83	131	c	143	63		103	
84	132	d	144	64		124	
85	133	e	145	65		105	
86	134	f	146	66		106	
87	135	g	147	67		127	
88	136	h	150	68		130	
89	137	i	151	69		111	
8A	138						
8B	139						
8C	140						
8D	141						
8E	142						
8F	143						
90	144						
91	145	j	152	6A		112	
92	146	k	153	6B		133	
93	147	l	154	6C		114	
94	148	m	155	6D		135	
95	149	n	156	6E		136	
96	150	o	157	6F		117	
97	151	p	160	70		160	
98	152	q	161	71		141	
99	153	r	162	72		142	
9A	154						
9B	155						
9C	156						
9D	157						
9E	158						
9F	159						
A0	160						
A1	161						
A2	162	s	163	73		163	
A3	163	t	164	74		144	
A4	164	u	165	75		165	
A5	165	v	166	76		166	
A6	166	w	167	77		147	
A7	167	x	168	78		150	

EBCDIC		GRAPHIC	ASCII		FLEXOWRITER	
Hex	Decimal		Octal	Hex	UC	LC
A8	168	y	169	79		171
A9	169	z	170	7A		172
AA	170					
AB	171					
AC	172					
AD	173					
AE	174					
AF	175					
B0	176					
B1	177					
B2	178					
B3	179					
B4	180					
B5	181					
B6	182					
B7	183					
B8	184					
B9	185					
BA	186					
BB	187					
BC	188					
BD	189					
BE	190					
BF	191					
C0	192					
C1	193	A	101	41	121	
C2	194	B	102	42	122	
C3	195	C	103	43	103	
C4	196	D	104	44	124	
C5	197	E	105	45	105	
C6	198	F	106	46	106	
C7	199	G	107	47	127	
C8	200	H	110	48	130	
C9	201	I	111	49	111	
CA	202					
CB	203					
CC	204					
CD	205					
CE	206					
CF	207					
D0	208					
D1	209	J	112	4A	112	
D2	210	K	113	4B	133	
D3	211	L	114	4C	114	
D4	212	M	115	4D	135	
D5	213	N	116	4E	136	
D6	214	O	117	4F	117	
D7	215	P	120	50	160	
D8	216	Q	121	51	141	
D9	217	R	122	52	142	

EBCDIC		GRAPHIC	ASCII		FLEXOWRITER	
Hex	Decimal		Octal	Hex	UC	LC
DA	218					
DB	219					
DC	220					
DD	221					
DE	222					
DF	223					
E0	224					
E1	225					
E2	226	S	123	53	163	
E3	227	T	124	54	144	
E4	228	U	125	55	165	
E5	229	V	126	56	166	
E6	230	W	127	57	147	
E7	231	X	130	58	150	
E8	232	Y	131	59	171	
E9	233	Z	132	5A	172	
EA	234					
EB	235					
EC	236					
ED	237					
EE	238					
EF	239					
F0	240	0	060	30	040	
F1	241	1	061	31	061	
F2	242	2	062	32	062	
F3	243	3	063	33	043	
F4	244	4	064	34	064	
F5	245	5	065	35	045	
F6	246	6	066	36	046	
F7	247	7	067	37	067	
F8	248	8	070	38	070	
F9	249	9	071	39	051	
FA	250					
FB	251					
FC	252					
FD	253					
FE	254					
FF	255					

APPENDIX B

STANDARD ESCAPE COMBINATIONS

@A	&	@&	^
@B	backspace		
@C	delete character	@+	v
@D	set device characteristics (see App D)		
@E	!	@<eol>	concatenate next line
@F	circumflex		
@G	grave accent		
@H	#	@/	\
@I	console attention (break) (Phoenix consoles only)	@(	[
@K	?	@)	]
@L	delete line	@,	;
@N	null	@.	:
@P	%	@$	£
		@-	_
@R	Reset case (See note 5)		
@S	$	@'	"
@T	¬ (or 'twiddle')	@<	←
@U	Upper case	@>	↑
@V	|	@@	@
@W	X'08' ('Graphic escape') see App. D		
@Xnn	hex input		
@Y	Lower case		
@Z	Reverses reflection (see note 6) (Phoenix consoles only)		

NOTES

1. Lower case letters are equivalent to upper case in escape combinations.
2. Escape combinations are always evaluated AFTER line-imaging and removal of erase characters, where appropriate.
3. 7-track flexowriters have three non-standard graphics, viz.

   @ is typed as X
   $ is typed as §
   ← is typed as →
4. @C removes entirely the preceding character in the line (if any). Processing of escape characters always takes place from left to right; hence escape combinations preceding @C count as a single character. For example, the sequence 'P@A@P@C@C' becomes 'P'.
5. When @R is used subsequent letters are left in their original case.
6. When @Z is used, no characters input are reflected; however, fault characters are still reflected as 'bell' and 'return' as 'carriage return, line feed'. Reflection is switched on again after another @Z or a carriage return whether at the end of an input record or following '@' or after '@L'.
7. See Appendix D for further details of console-dependent escape combinations

APPENDIX C

PRINTER CHAIN

The printer chain contains the following characters:

```
1234567890.-();ABCDEFGHIJKLMNOPQR&STUVWXYZ,=*<>'1234567890.-

@#£ABCDEF+abcdefghijklmnopqrstuvwxyz1234567890.-();ABCDEFGHI

JKLMNOPQR/STUVWXYZ,=*<>'1234567890.-$*%\←↑|:¬+abcdefghijklmn

opqrstuvwxyz1234567890.-"^V ABCDEFGHIJKLMNOPQR/STUVWXYZ[]_!?*
```

The speed of printing of a character depends on its number of appearances on the chain. The nominal speeds for whole lines containing only characters with the same frequency of appearance vary from 1100 lines per minute (5 appearances) to 310 lines per minute (one appearance). For mixed lines the speed is largely governed by the characters on the line with the least frequency of appearance on the chain. This should be borne in mind by users when planning which special characters to use for a particular printing purpose. In particular it should be noted that - . and * appear more frequently than any other non-alphanumeric character.

The following table gives characters by frequency of appearance:

5 appearances	(1110 l.p.m.)	0 to 9 . -
4 appearances	(995 l.p.m.)	A to F *
3 appearances	(775 l.p.m.)	G to Z
2 appearances	(560 l.p.m.)	a to z = ( ) / + ' , ; < >
1 appearance	(310 l.p.m.)	£ $ @ | : % ¬ ? # [ ] ↑ &
		" ! ← _ \ ^ V

APPENDIX C

PRINTER CHAIN

The printer chains on the IBM 1403 printers at the central site contain the following characters:

1234567890.-();ABCDEFGHIJKLMNOPQR&STUVWXYZ,=*<>'1234567890.-

@#£ABCDEF+abcdefghijklmnopqrstuvwxyz1234567890.-();ABCDEFGHI

JKLMNOPQR/STUVWXYZ,=*<>'1234567890.-$*%\ ←↑|:¬+abcdefghijklmn

opqrstuvwxyz1234567890.-"^\/ABCDEFGHIJKLMNOPQR/STUVWXYZ[]_!?*

The speed of printing of a character depends on its number of appearances on the chain. The nominal speeds for whole lines containing only characters with the same frequency of appearance vary from 1100 lines per minute (5 appearances) to 310 lines per minute (one appearance). For mixed lines the speed is largely governed by the characters on the line with the least frequency of appearance on the chain. This should be borne in mind by users when planning which special characters to use for a particular printing purpose. In particular it should be noted that - . and * appear more frequently than any other non-alphanumeric character.

The following table gives characters by frequency of appearance:

5 appearances	(1110 l.p.m.)	0 to 9 . -
4 appearances	(995 l.p.m.)	A to F *
3 appearances	(775 l.p.m.)	G to Z
2 appearances	(560 l.p.m.)	a to z = ( ) / + ' , ; < >
1 appearance	(310 l.p.m.)	£ $ @ | : % ¬ ? # [ ] ↑ & " ! ← _ \ ^ \/

The MDS 4340 printers at the Sidgwick Avenue, Chemistry and DAMTP outstations are chain printers with a nominal speed of 600 lines per minute. The character set is the same as that for the IBM 1403 printers, but each character appears 4 times on the chain.

The DP2260 printers at the New Addenbrookes and West Cambridge outstations are drum printers with a nominal speed of 436 lines per minute. As all characters appear on the drum in each print position, printing speed is not affected by the text being printed.

The character set is similar to that on the IBM 1403 printers except that

£ is replaced by ¢

↑ " " " →

∨ " " " Δ

APPENDIX D

PHOENIX ON-LINE INTERFACE

The PDP-11 multiple access software supports a number of devices with differing characteristics. However, since the majority of connected devices are Standard Teletypes, a description of their interface is given first, followed by descriptions of how the interfaces for other devices differ.

1. STANDARD TELETYPES

These are 64-character ASCII Teletypes, operating in full duplex mode. This means that all printing is under the control of the computer, including the reflection of characters typed by the user. If no reflection appears when a character is typed, it indicates one or more of the following conditions:

(a) The Teletype is logged off, and will only respond to 'return'.

(b) The PDP-11 is stopped.

(c) The PDP-11 has run out of buffers (hopefully rare).

(d) The current input line has exceeded 256 characters in length. The line can be cancelled by @L.

(e) The 370/165 has stopped issuing read requests to the PDP, because it has run out of buffers for the user. This condition clears when the user's running program accepts some of the stacked up input, or if 'break' is pressed or @I typed. (See Section 4.11.1)

(f) The 370/165 is stopped.

State (f) changes to state (a) if the 370/165 hardware is reset; the message '***ABANDONED' is sent to any terminals that were logged on.

Logging on

Pressing 'return' on a logged-off Teletype causes one of the following messages to be typed:

(a) PHOENIX NOT AVAILABLE

(b) Operator supplied alternative to (a),
e.g. 370 CPU BROKEN - ENGINEER CALLED

(c) ONLINE SYSTEM FULL

(d) LOGON

After (d) the user should type his identifier terminated by 'return' or 'ESC'.

Mistakes can be corrected by @C or @L (see later), but to abandon the logon attempt altogether, 'break' must be pressed; in the latter case, ***ABANDONED is typed. This message also appears if Phoenix is stopped during LOGON, or if the identifier and 'return' are not typed within 30 seconds.

Input

(a) The following characters reflect themselves and are treated as ordinary data characters, to be translated to EBCDIC code and passed to the 370/165 when the input record is complete:

A - Z
a - z (not available on keyboard, but could be on paper tape)
0 - 9
space
[].<(+&!$*);-/\,%>?:#'="

(b) Circumflex or ↑ (ASCII 136) will input ↑ (EBCDIC 65)
Underline or ← (ASCII 137) will input ← (EBCDIC 66)

(c) '@' is an escape character, the character following it is treated specially (see below).

(d) 'Tab' reflects 'space', but transmits 'tab' to the 370/165.

(e) 'Rubout' is converted to (and reflects) the character pair '@C' (see below).

(f) 'Cntrl/Y', 'cntrl/U', 'cntrl/R', 'cntrl/Z' reflect themselves (i.e. do not print), but are treated as '@Y', '@U','@R'and '@Z' (see below).

(g) 'Return' causes 'carriage return, linefeed' to be reflected, 'carriage return' to be sent to the 370/165, and terminates the input record.

(h) 'Esc' reflects itself, sends 'esc' to the 370/165, and terminates the input record.

(i) 'Null' (ASCII 0) is completely ignored (paper tape runout).

(j) All other characters are reflected followed by 'bell' as a warning that the character is 'illegal'. Neither the illegal character nor the 'bell' are sent to the 370/165.

(k) The 'break' key causes the input record to be cancelled, and a 'break' to be sent to the 370/165 (this is described in Section 4.11.1).

Escapes

The character '@' is an escape character, which causes the character(s) following it to be treated specially. Most escape combinations consist of '@' followed by a single character, though there are some which require more. Upper and lower case letters are treated as synonymous in escape combinations. If 'N' is typed as any one of the escaped characters (except after @W), the escape combination is immediately abandoned. If 'return' is typed as any one of the escaped characters (except after @W), the escape combination is immediately abandoned, 'carriage return, linefeed' is reflected, but the input record is not terminated (and nothing is sent to the 370/165). If an illegal character is typed after '@', 'bell' is reflected, the character is ignored, and the system remains in 'escape' state for the next character.

Escape characters which translate to other characters are:

@A	&	@>	↑
@B	backspace	@&	∧
@E	!	@+	∨
@F	circumflex	@@	@
@G	grave accent	@/	
@H	#	@(	[
@K	?	@)	]
@P	%	@,	;
@S	$	@$	£
@T	¬	@-	
@V	|	@<	←
@W<char>	X'08' ('Graphic escape') followed by the untranslated character	@'	"
@Xnn	EBCDIC X'nn'	@.	:

The @W mechanism is intended as a means of getting control characters into the 370/165 in a form suitable for subsequent output (see below). '@W<char>' is precisely equivalent to '@X08@XCC', where CC is the hexadecimal expression of the ASCII character.

Escape combinations which will perform control functions are:

@<return> Reflects 'carriage return, linefeed', but does not terminate the input record, or send any characters to the 370/165.

@C Cancels the latest character in the buffer which is to be sent to the 370/165 (thus '@A@CE' becomes 'E', not '@E').

@Dc Select device option c (a single character). This combination is used to tell the PDP-11 about characteristics of an ASCII terminal which are different from the Standard Teletype, and also to select optional features of the interface. Fixed lines are set up with the correct characteristics, so the first type of combination is normally only needed on dial-up lines. The following options are available, those recognised on IBM 2741 terminals being marked * :-

@DB	Ignore BREAK character - @I can be used instead SET TERMINAL NOBREAK is equivalent to @DB.
@DD *	Default (reset to Standard) SET TERMINAL DEFAULT is equivalent to @DD.

@DF	Full 96-character ASCII character set SET TERMINAL FULLCHAR is equivalent to @DF.
@DH	Half-duplex device; SET TERMINAL HALFDUP and SET TERMINAL HALFDUPL are synonyms.
@DI *	Invert the case of all subsequent input letters; SET TERMINAL SWCASE is equivalent to @DI.
@DJ *	Cancel a previous @DI. SET TERMINAL - SWCASE is equivalent to @DJ.
@DN *	No action, the @D was a mistake.
@DP	Rubout reflects deleted characters. @DP is described in Section 2. SET TERMINAL VERIFY is equivalent to @DP.
@DQ	Rubout reflects 'backspace space backspace'. @DQ is described in Section 2. SET TERMINAL VDU is equivalent to @DQ.
@DR	Ignore RUBOUT character - @C can be used instead; SET TERMINAL NORUBOUT is equivalent to @DP
@DS) @DT) @DU) @DV)	Slow carriage return - see below SET TERMINAL IDLES SET TERMINAL IDLET SET TERMINAL IDLEU SET TERMINAL IDLEV are equivalent
@D<return>	As @<return>

@DB and @DR are useful for poor quality dialled lines to avoid noise generated RUBOUT and BREAK.

The changes in interface for non-standard ASCII devices are described in later sections.

@I	Interrupt -- has the same effect as pressing 'break'
@L	Cancel the whole input record; 'carriage return, linefeed' is reflected after this combination
@N	Null, the @ was a mistake
@R	Reset case -- subsequent letters are left in their original case (subject to @DI)
@U	Upper case -- subsequent letters are forced to upper case
@Y	Lower case -- subsequent letters are forced to lower case
@Z	Reverse reflection of input characters temporarily.

There is an implicit @R and reflection is switched on again after a carriage return, whether at the end of the input record or following @ or system-generated after @L.

@U and @Y act <u>after</u> any case switching caused by @DI. Letters are always reflected in the case in which they are to be sent to the 370/165.

When in '@Z' state, fault characters still reflect themselves followed by 'bell', and 'return' still reflects 'carriage return, linefeed'.

Output

(a) The following characters are translated to ASCII code but otherwise output unchanged:

A - Z
a - z
0 - 9
space
[].<(+&!$*);-/'\,%_?#@'="
carriage return
linefeed
esc
bell

(b) 'Tab' is converted to 'space'

(c) 'Newline' is converted to 'carriage return linefeed'.

(d) EBCDIC 'null' (X'00') is ignored

(e) EBCDIC ↑ (65) is output as ASCII 136 (^ or circumflex)
EBCDIC ← (66) is output as ASCII 137 (← or underline)

(f) EBCDIC X'08' ('Graphic escape) causes the following character to be output untranslated. This mechanism is intended as a means of sending various control characters to vdu terminals.

(g) All other characters are converted to '?'.

Break-in

Whenever the PDP-11 finishes transmitting a record to the 370/165, it starts accepting input characters again There is thus only a short period after the user types return (or esc') during which input is not accepted.

When output for the terminal arrives from the 370/165, the PDP-11 behaves as follows:-

a) If the user has not typed any valid characters since the end of the last input or output line, the PDP-11 stops accepting input, and proceeds immediately to the output.

b) If the user has typed valid characters, the PDP-11 allows 60 seconds for the completion of the input record. If this deadline is met, the input is sent off to the 370/165 while the output is taking place. Otherwise '!! carriage return, linefeed' is output, and the incomplete input record is abandoned.

Interruption of output

Input is not accepted while output is taking place, but there are two signals which can be sent to the PDP-11:

(a) 'Break' is sent by typing 'break' or '@I' (not reflected)

(b) 'Pause' is sent by typing 'space'

'Break' causes the rest of the current output record and any outstanding input record to be lost, and a "break" signal to be sent to the 370/165 (See Section 4.11.1). 'Pause' is purely a local PDP-11 feature which is intended for use on Teletype-emulating VDU's. It causes the PDP to pause at the end of the current output record. The pause lasts 60 seconds. The pause can be cancelled by typing '@L'. Alternatively, a line of input may be typed. This line is sent to the 370/165 while subsequent output proceeds.

2. NON-STANDARD ASCII DEVICE CHARACTERISTICS

96-character ASCII

The only differences from the default are in character translations:

(a) Circumflex or ↑ (ASCII 136) corresponds to EBCDIC 6A (circumflex)
Underline or ← (ASCII 137) corresponds to EBCDIC 6D (underline)

(b) Curly brackets, vertical bar, not sign (twiddle), grave accent and backspace are recognized in input and output.

(c) EBCDIC 65 (↑) and 66 (←) are output as fault ('?').

(d) Vertical tab and formfeed cause linefeed to be input, and are permitted in output.

(e) The graphics control characters SO, SI, GS and US are recognised in input and output.

(f) 'Delete' (X'07') is output as 'rubout' (ASCII 377).

(g) 'Linefeed' is reflected and transmitted.

Half duplex

A half duplex device does its own printing of input characters, so the PDP-11 has no control over the reflection of characters. It does, however, output 'bell' when an illegal input character is received, and 'idle, linefeed' after 'return'. If the device has a slow carriage return, further idles may be output.

Slow carriage return

Some devices need more than two character times for the mechanical functions of returning the carriage to take place. For these devices, the action taken depends on three numbers

(a) a fixed number of 'idle' characters to be inserted after carriage return or newline

(b) the number of character positions for which an extra 'idle' is required after carriage return

(c) a fixed number of 'idles' to be inserted after linefeed

In addition for ASCII devices, if a carriage return is not followed by a linefeed, an extra idle character is inserted. The idle character used for ASCII devices is zero.

The defaults are:

	a	b	c
ASCII devices	0	32767	0
2741 terminals	2	10	1

Other slow return standards are

a	b	c	
2	32767	0	corresponding to @DS
3	10	0	corresponding to @DT
4	32767	0	corresponding to @DU
0	15	1	corresponding to @DV

Other sets of numbers, corresponding to @DW, @DX ... etc. may be added as necessary.

3. IBM 2741 TERMINALS

The IBM 2741 is the only non-ASCII terminal supported by the system. It is a half duplex device which operates strictly in either 'transmit' or 'receive' state. In the latter state the keyboard is locked, while in the former state the terminal ignores all characters sent by the computer. The 'transmit interrupt' feature does allow the computer to interrupt input from the terminal, but use of this is cumbersome (it involves sending a 'break' to the device). For this reason no character-by-character interaction is implemented. Note that the 'Attn' key has a different effect, depending on whether the device is in 'receive' or 'transmit' state.

Input

(a) The following characters are translated to EBCDIC code and passed directly to the 370/165:

A - Z
a - z
0 - 9
space
tab
backspace
-#.&/$,_£*:<="¬>+)'?(%;!|

(b) '@' is an escape character (see below).

(c) 'Return' causes a carriage return and linefeed, and terminates the input record and sends 'carriage return' to the 370/165.

(d) 'Attn' terminates the input record without moving the carriage and sends 'esc' to the 370/165.

(e) There are no illegal single characters on a 2741 keyboard, but if any illegal character is received it is converted to '?'.

Escapes

The escape combinations are the same as for ASCII devices, with the following provisions:

1. It is not possible to inform the user of bad escape combinations at the time of typing; they are translated to '?'.

2. @W, @Z and those @D combinations which set device characteristics are ignored.

Output

(a) The following characters are translated to 2741 code, but otherwise output unchanged:

A - Z
a - z
0 - 9
space
tab
backspace
newline
-@#.&/$,_£*:<="¬>+)'?(%;!|

(b) The combination of one or more carriage returns immediately followed by linefeed is output as 'newline'. Isolated linefeeds are output as 'linefeed'; isolated carriage returns are replaced by a sufficient number of backspaces to get the carriage to the left hand margin.

(c) All other characters are converted to "? .

(d) While output is taking place, the keyboard is locked except for the 'attn' key, which, if pressed, causes the output to terminate and 'break' to be sent to the 370/165.

4. LINE SPEEDS

Permanently connected lines may be set up as required, normal speeds are 110 baud, 134.5 baud, 300 baud and 1200 baud; unless otherwise requested 110 baud lines are set up as 64 character ASCII (teletype), 134.5 baud lines as IBM 2741 terminals. 300 baud and 1200 baud lines as 96 character ASCII. The characteristics of ASCII devices may be changed using the escape mechanism (see above).

Dial-up line speeds are automatically detected when the user types 'return The following speeds may be used and have the indicated default characteristics assigned:

110 baud	64 character ASCII (teletype)
134.5 baud	2741 terminals
150 baud)	
200 baud)	96 character ASCII
300 baud)	

Again the assigned default characteristics of ASCII devices may be changed by the escape mechanism.

5. DIALLING UP

The modems on Cambridge 66401 are automatically answered, and provided the PDP-11 is running, ringing tone should change to data tone (a high pitched whine) within two or three seconds. Following this, the PDP-11 waits 10 seconds (to allow the user to connect an acoustic coupler), and then it begins to monitor the 'carrier detect' line from the modem. If at any time the carrier is found to be absent, the PDP hangs up and the user is logged off. The PDP-11 also hangs up if the user has not logged on to Phoenix within 60 seconds.

6. ERROR CONDITIONS

If the PDP-11 has to be restarted, the message ***ERROR is output; the current input line is lost and must be retyped. If output was in progress it may be repeated.

If an (unexpected) IPL occurs in the middle of an online session ***ABANDONED is output.

If a terminal remains unattended for 5 minutes (i.e. there is no associated 370/165 activity) a warning is output by the 370/165; after a further 5 minutes the session is cancelled and the user is logged off.

7. SPONTANEOUS MESSAGES

1. Whenever the Phoenix online system is started for computing service running, the message PHOENIX STARTING is sent to all connected consoles.

2. Whenever a new operator message is put into the communications system (usually giving details of why Phoenix is not available) it is sent to all connected consoles.

8. UPPER AND LOWER CASE LETTERS

The escape combinations @Y, @U and @R, and the devices option @DI are provided to facilitate the input of both cases of letter on all types of terminal.

@Y and @U (or cntrl/Y and cntrl/U on ASCII devices) force the case of all subsequent letters until the end of the current line (or another @Y, @U or @R). @R cancels a previous @Y or @U on the same line.

@DI causes the case of all letters to be inverted, before the application of any forcing due to @Y or @U. This has two main applications:-

a) On consoles with upper case keyboards only (e.g. Teletypes) it has the effect of resetting to lower case at the start of each input line, instead of the default upper case. Subsequent @Y's and @U's can be used to force cases, and @R resets to the initial state, i.e. lower case.

b) On consoles with two-case keyboards @DI reverses the sense of the shift key. This can be useful when typing in a mixture of upper case and letters and digits, for example.

On full duplex devices, the case transformations are carried out before reflection takes place, so that letters are always reflected in the case in which they are to be sent to the 370/165.

Case transformations are not applied to letters which are part of an escape combination (where upper and lower case are synonymous); these are reflected unchanged.

9. BREAK SIGNALS

Once the user has logged on to Phoenix, a 'break' (attention) signal can be generated at any time:-

a) On an ASCII device by pressing 'break' or typing @I. If the PDP-11 is not accepting input, @I will not get reflected (to full duplex devices).

b) On an IBM 2741 terminal by pressing 'attn' if the terminal is typing, or by typing @I if the keyboard is unlocked. Note that pressing 'attn' when the keyboard is unlocked does not generate 'break', but terminates the input line with 'esc'.

On receipt of 'break' the PDP-11 abandons any outstanding input and output that it may have, sends a 'break' signal to the 370/165, and types !<return on the user's console. Further 'breaks' are ignored until the first is acknowledged by the 370/165. This normally happens immediately. The effect of break on programs running in the 370/165 is described in section 4.11.1.

APPENDIX E

PHOENIX MACRO COMMANDS

Macro commands are used by Phoenix to expand simple commands into the necessary basic commands; for example

```
COMMAND * A=B  becomes %%C ( %COMMAND=*) A=B
```

This document describes the construction and use of a Phoenix macro command library.

1. NOMENCLATURE

Phoenix macro commands compile into an interpreted form. The element of Phoenix that interprets these compiled macros is the Phoenix Macro Decoder. Both macro commands and compiled macros are referred to as 'macro's; if it is necessary to make the distinction there is a macro source, and a macro binary. A collection of macros form a (macro) library.

2. MACRO SOURCE CODE

Command macros are written in the Assembler macro language and compiled by the Assembler. Most of the peculiarities and nasties are directly attributable to the nature of the BAL macro language; nevertheless it has proved a very easy vehicle in which to write new macro commands quickly.

The data for the assembler macro consists of one positional parameter with subfields delimited by the brackets surrounding the data, and interior comma's. These subfields are the elements of the 'Phoenix macro command' language. Continuations and use of ' and &, follow normal Assembler rules.

The subfields are interpreted in triples, each group of three forming one directive to the Decoder.

The macro, which is called CMNDPROF, is thus written:

```
name CMNDPROF  ( V1,V2,V3,                              x
                 V4,V5,V6,                              x
                 - -                                    x
                 V3n-2, V3n-1, V3n)
```

where V3r-2 V3r-1 V3r is the rth triple.

Example

Before describing in complete detail the syntax and semantics of macro commands, it will be helpful to consider an example. The source of the main Phoenix macro library is BL10.SOURCE:PROF and this example is taken from there.

```
1          EDSORT CMNDPROF (':EDSORT(',FLAG,EKET,  x
2                    K,,'OLD',                     x
3                    '/R OLDED=',DSN,'$',          x
4                    K,,'NEW',                     x
5                    '/R NEWED=',DSN, '%C',        x
6                    K,,'TO',                      x
7                    '/N MERGED=',DSN,'%O',        x
8                    END)
```

Interpreting this in triples as indicated, we see that lines 2,4,6 all start with 'K' . This is short for KEY and indicates that the item defines a key, whose name is the third component of the triple. (The second component, unused in this example, qualifies the type of key). The items following a key up to the next, define the action to be taken for that key. The last item, line 8, defines the end, and need not be a complete triple. Items before the first key item, line 1 in this case, form POSITIONAL items which are obeyed by the Decoder searching for keys.

Data items (lines 1, 3, 5, 7 above) have two text values (1st and 3rd parts of triple) separated by a verb. In this case the only verbs used are FLAG and DSN,

DSN means read a dataset name
- this can also be a sequence of dsds each separated by a comma or a '+'

FLAG means set the flag indicated in the third part, in this case indicating:- end entire line with)

The first text is always output when the item is 'obeyed'; the Decoder then obeys the verb, and if necessary uses the third item as default text.

When the Decoder detects the end of the line being read it substitutes defaults for all unused keys. Thus the above example instructs the Decoder to:

1. Output :EDSORT(

2. Set a flag ensuring a ket at the end

3. Search for keys (no more positional items)

4. If key=OLD output /R OLDED=
   then dsn (or $ if no dsn found)

5. If key=NEW output /R NEWED=
   then dsn (or %C if no dsn found)

6. If key=TO output /N MERGED=
   then dsn (or %O if no dsn found)

7. Then scan for unused keys and fill in defaults /R OLDED=$ etc.

8. Output closing) [because of 2]

Additionally the semantics indicate:

If when searching for a key in the command line the next thing read is not a key (i.e. not a word ending in = or space) and no keys have been read, then assume the data is for the next key item, with the key omitted.

Thus

EDSORT OLD .A and EDSORT .A both give

:EDSORT(/R OLDED=.A /R NEWED=%C /N MERGED=%O)

While EDSORT TO .B OLD .A gives

:EDSORT(/N MERGED=.B /R OLDED=.A /R NEWED=%C)

Note change of order.

3. SYNTAX

Rules for Command Decoder

WORD — Item of up to 8 characters, first is letter or national, remainder letters.

INT — Item of up to 8 digits.

NAME — Item of up to 8 characters, first is letter or national, remainder letters or digits.

TEXT — Arbitrary string; if surrounded by quotes the exterior quotes are removed and interior double quotes become single quotes; if the string does not start with quote, then it ends on space.

QTEXT — As TEXT except that the quote stripping does not take place.

REST — Remainder of line; or quoted text; if surrounded by quotes, the quotes are stripped off as for TEXT.

QREST — As REST except that the quote stripping does not take place.

DSN — Character string that could form set of dsd's. I.e. first character may be ., +, national or letter with remainder anv characters, ending in space.

ALL — String that is terminated by end of line.

Additional rules for the command decoder

1. Whenever a quoted string may occur in a data triple a piece of text starting with parameter-occurrence may be used instead. If spaces are required, the parameter-occurrence should take the form

   <'parameter-name$default>other text'

2. A KEY is a WORD terminated by = , space or newline or the value of a macro parameter.

   Thus FROM FROM= are keys

   FROM.A
   FROM1 are not

Pseudo Quotes

The characters 28 and 29 hex (represented by ↓ and ↑ respectively) are used in Phoenix decoding to delimit strings. They are treated as follows:

When decoding a macro any items read by REST, QREST, TEXT, QTEXT or ALL will be delimited by a ↓ ↑ pair. If such an item is assigned to a * variable the ↓ ↑ will be omitted. When reading text, i.e. in macro decoding using REST, QREST, TEXT, ALL or QTEXT or when reading a parm field after %D or), if the first character is ↓ then the string will end on a matching ↑ and the outermost ↓ ↑ will be removed. This action also applies when reading the value of a parameter either in the PARS (SET P) command or in the argument string of a C-command; so that

   PARS A=↓B C D↑
   sets parameter a to the value B C D

Otherwise ↓ and ↑ will be skipped whenever a 'search for the next significant character' takes place, so that for example

   EDIT ↓.A is the same as EDIT .A

Finally, when passing a parm string to a command any ↓ or ↑ characters are removed.

Rules for BAL Macro Assembler

'&' must always be '&&' (even inside ' ')

Inside ' ' '' = '

Otherwise inside quotes anything is allowed. Not more than 255 characters are allowed. Continuations: Non blank character in column 72; next line starts in column 16.

Full description of the Macro source code

The macro source code consists of triples, T1, T2 and T3.
There are two types of triple - data triples and directive triples.

4. DATA TRIPLES

These control the reading of the data line.

T1: may be either of a) 'text'
or b) *n (where n is a single digit)

a) Provides text to be output before the item

b) Is a directive indicating that macro parameter n is to be set to the result of the triple.

Note that if text is the empty string then the quotes should be omitted.

T3: As T1; provides a default value for either 'text' or *n
*0 is initialised to the command name.

T2: Is an action verb - one of:

NAME	)	which instruct the decoder to
INT	)	read an item of requisite type.
WORD	)	
TEXT	)	
REST	)	
DSN	)	
QTEXT	)	
QREST	)	
ALL	)	

END	signifies the final item of the macro.
Empty or NULL	no action (other than copying text T1).
NULLEKET or NK	NULL + set 'end with ket' flag, which causes a) to be inserted at the end.
EQ	no action other than copying texts T1, T3.
FLAG or F	set flag indicated in T3; flags available are:-
	EKET or EK - set 'end with ket'
	UC - set upper case.
	NUC - cancel upper case.

If T1 = *n; NULL no setting

EQ set *n to text T3

If T3 = *n; The value of *n is used

Thus *1, EQ, *2 sets *1 equal to the value of *2

while 'A', EQ, *2 outputs the text A<*2> (where <*2> implies the contents of *2).

5. DIRECTIVE TRIPLES

These supply directives to the decoder.

T1 = Directive type; which is a word

END	An end marker is compiled; this should be the final item
KEY or K	Keyword item T2 = key type (see below) T3 = key word (in quotes) or variable. 'WORD'
KEYNEWP	T2 = address T3 = keyword This directive defines a key that if found causes the decoder to restart the data line using the macro at address T2.
JUMP LINK	T2 = address T3 = ignored The decoder carries on decoding at address T2, which must be another directive. If the directive is LINK the address of the next triple is preserved.
RET[URN]	This should be the final item - the decoder returns to the last LINK. RET = END if there have been no LINKS. N.B. 1. LINK's may not be nested. 2. JUMP's, LINK's, and RETURN can be used to evade the 255 character rule.

Key Types

empty = OPT	The key and the item are optional.
NOTOPT	A value must be supplied for this item; the key is optional.
OPTNA	The item is optional, but if present must be keyed.
NA	NOTOPT + item must be keyed.
NODEF	Optional item if omitted no default values are substituted.
OPTNAND	The item is optional; if omitted no default values are substituted. Must be keyed if present.

6. MACRO LIBRARIES

These are load modules containing assembled and linked macros. They have a heading of the form

0-3	TABLE PTR	Position of name table
4	FLAG BYTE	see below
5-7	LENGTH	Length of the name table (bytes)

Flag byte: d7=1 name table is sorted, and will be searched using an efficient search method.

Elsewhere in the module, word aligned, is the

Name table - 12 byte entries

NAME (8)	POINTER

The name is left justified and blank filled; the pointer points to the macro.

The Macros form the rest of the module.

The canonical form thus is:

0	P = 8	
4	n*12	
8	Name 1	Macro 1 pointer
20	Name 2	Macro 2
32	∿	∿
	∿	∿
	Name n	Macro n

Macro 1 -

Macro 2 -

Note that two or more name items can point to the same macro, in which case they are synonyms.

7. SEMANTICS, OR, RULES OF THE DECODER

All items before the first directive are positional; the items up to the first directive are performed before any key decoding.

Data items following a KEY item describe the actions to be taken for that KEY. If a KEY item immediately follows another KEY item then the keys are synonyms (this can be repeated indefinitely).

When searching the KEYS of a macro JUMPs LINKs and RETURNs are obeyed, and otherwise items are scanned key by key (which are chained together for efficiency).

Decoder Actions

1) Do all POSITIONAL actions

2) Note position of first directive (D)

   Set 'No Keys matched' (K = 0)

3) Read key from a command line:- if next item is key go to 5);

   else go to 11) if end line, else

4) error if K = 1

   scan for first unused keyed item
   - error if none

   error if key is NA

   go to 6)

5) Scan for first key which matches this key (starting at D)
   If not found, go to 4), resetting pointers to reread the word
   If found set K = 1 and

6) Mark the keyed item used - error if already used

7) If next item = key (no data items) go to 6)

8) Output space (to separate keyed items)

9) Do data items

10) Go to 3)

11) At line end, scan for unused keys

    For each found i) Error if NOTOPT or NA [item omitted]

    ii) Do data items (using defaults) unless NODEF

12) Add) if EKET switch set.

APPENDIX F

The Cambridge 370/165 Scheduling System

Objectives

The objectives of the Scheduling System are fourfold. First and foremost it aims to share out use of the available resources according to allocations determined by the management of the Computing Service in consultation with the user and his institution. Thus the system is concerned with regulating the rate at which resources are consumed by each authorised project as a proportion of the total available; it does not work in terms of fixed and absolute amounts of machine time. Next, the system aims to maximise the utilisation of the whole machine, in particular by ensuring that a 'good mix' of different types of work is processed simultaneously. The third objective is to provide users with a reasonable prediction of when their jobs will be run; the combination of a need to maximise overall system utilisation, the inherent fluctuations in user demand, and the inevitable occurrence of breakdowns from time to time means that predictions can never be perfect. Finally, an attempt is made to tailor the turnround and resource usage to users' requirements, and in particular to provide a high degree of immediate or 'while you wait' turnround.

Since these objectives are to some extent contradictory, the Scheduling System is unable to achieve perfection in any one of them. Thus the system has been developed, and is constantly being re-tuned, to obtain a reasonable balance between them.

Definitions and Terminology

User identifiers

In most cases each individual user is assigned a unique user identifier (see Section 2.4.1). This identifies the person for whom work is being done and any output produced. Certain privileges granted by the management are associated with user identifiers (e.g. use of BIGDISC, private disc packs, etc.).

Projects

Individual allocations of resources are assigned to project numbers, and the management can arrange for one or more user identifiers to be associated with each project. It is also possible in appropriate circumstances for a user identifier to be associated with more than one project number. All work done on the system is controlled by the resources allocated to a project and is accounted to that project. Thus, all jobs submitted for processing must quote both a user identifier and a project number and the system checks that the combination of these is authorised.

Certain quantities are held by the system for each project number, the most important being:

Shares (S) the number of shares allocated by the management; it is a measure when compared with the total active shares issued to all projects of the proportion of available machine resources permitted.

Usage (U) a measure of the current rate of working on the project; as resources are used, U is incremented by the amount of resources consumed multiplied by the priority. It also decays with time (further details are given below).

Outstanding Work (W) it records at any time the estimated total resources multiplied by the priority for work waiting in the machine to be processed. It is used for scheduling purposes in combination with U. W is incremented when a job is read into the system and decremented at the end of the job.

Rate of Working (V) a measure of the rate of working per share. Its value will normally be in the range of 0 and 10 with the mode over all projects being less than 1. It indicates the extent to which the project is getting value for its shares.

Jobs

All jobs submitted to the system are placed in a job queue (typically a few hundred jobs long) in a position depending on various scheduling parameters. These are:

(a) the project's shares and current usage as defined above,

(b) the stated requirements of the job which are given via TURNROUND and/or PRIORITY statements in the job description (see Section 2.5 and below),

(c) the size of the job given by the various JD statements (LIMSTORE, TIME, TAPE etc.),

(d) the current and anticipated load of other work on the system.

The following quantities are associated with each job:

Estimated Cost (Ce) calculated from the LIMSTORE and TIME estimates in the job description using the formula in Section 2.7.2.

Cost (C) the actual cost of the job according to the same formula with the store and cpu times actually used. This is calculated after the job has run.

Priority Limit (Pl) as given by the user (explicitly or by default) in the job description.

Priority (P) the priority value assigned by the Scheduling System in a manner described below.

Merit (M) the scheduling merit for the job determined by the Scheduling System as described on the following pages.

Whenever a job is run its system output includes the following values:

for the relevant project:	S (shares), U (usage excluding the current job), V (rate of working/share) W (outstanding work including the current job)
for the job:	P (priority), M (merit), and C (the resource units used)
for the system:	N (the value of M for the currently running jobs when the job was put in)

Immediately after a job has completed its output, the project's U is incremented by its charge, which is the cost (i.e. resource units used) multiplied by the priority P.

General features of the scheduling mechanism

The job queue maintained by the system is controlled by a continually running system program called HASP (Houston Asynchronous Spooling and Priority system). Jobs are inserted into this queue by the Scheduling System, the position in each case depending on a combination of factors, the chief ones being the user's requirements for turnround and the history maintained by the system of the total load imposed by all other jobs. Jobs are generally taken from the head of the job queue for execution under control of the main operating system (OS), the selection being made in such a way as to ensure a good 'job mix'. The Scheduling System keeps records of the actual rate at which jobs are processed which, together with information on the rate of job submission as a function of the time of day, enables predictions to be made about the running of jobs.

When a user submits a job the system endeavours to give the turnround requested, either explicitly or implicitly, in the job description.

The Scheduling System first attempts to satisfy this requested turnround by assigning a merit value (M) for the job (see below for details). If the requested turnround cannot be met, then the system tries again with any alternative turnround specified by the user. One important turnround is 'immediate' which can be given explicitly by TURNROUND NOW. The independent default values for the initial and the subsequent request are NOW and OVERNIGHT. The system may fail to provide the requested turnround if the overall load is too heavy and either the project's usage is too high or the user has specified a priority limit which is too low. Failing this attempt, the system schedules the job for the best achievable turnround with a priority equal to the specified priority limit. It should be noted that,

since the priority of a job affects the charge being made for it and since the value of priority as chosen by the system is bounded by the priority limit, the user can thereby limit this charge. The priority limit defaults to a value depending on the user's estimate of the size of the job (see below).

Thus the system will assign a turnround, the allotted turnround, for each job; this will either be FAST (denoting immediate turnround) or some period of time (minutes, hours or days). This allotted turnround appears in the output of the STATUS command and on control consoles of work station terminals. In the case of FAST this implies a turnround that should be within a few minutes; otherwise the time indicates the period that is expected to elapse before the job begins execution. The user should note that the actual time by which the job will have run is the allotted turnround (which is subject to the inherent uncertainties of the mechanism) plus the time taken for OS to process the job. The allotted turnround for a job is constantly kept up to date (one minute is subtracted for every one minute), but there is one major exception. If the system finds that its predictions for immediate turnround jobs have become seriously wrong (either due to an unexpected job load or as a result of unscheduled down-time), then jobs in the job queue are rescheduled. Allotted turnrounds will thus be lengthened by this process, and some jobs may be moved into the night-time period and others already scheduled for night-time running may have to be deferred for an additional 24 hours. The allotted priority is unchanged by this process.

By use of the STATUS command, users can keep a watch on the allotted turnround of a job and on its progress. If at any time this is considered unsatisfactory, the user is always able to kill the job using the KILLJOB command (see Part II), and resubmit it with a new requested turnround and/or priority limit.

The default values for the requested turnround (NOW or OVERNIGHT) and the priority limit (which depends on the size of the job) are chosen in an attempt to satisfy users whose rate of working is appropriate for the shares allocated and who have a wide job range. However, since there is so much variation both in users' individual needs and in the total load on the machine, most users except those whose computing needs are quite small will need (i) to know more about how the scheduling system works, (ii) to choose explicit turnround and, possibly, priority limits for their work so as to get their own relative priorities right while keeping their total rate of working under control, and (iii) to watch the merit and priority values chosen by the system together with their usage. How to do this appears below.

Details of the scheduling mechanism

This section first describes those details of the mechanism needed for more than very simple use of the system.

As mentioned earlier, whenever a job is submitted it is assigned both a merit value (M) and a priority value (P); how these are calculated is described below. For the moment it is sufficient to think of M as being a measure (relative to all other jobs) of how deserving of running the job is - the higher the better. P on the other hand reflects the cost of running the job, and it is used in the calculation of the charge to be made against the project's usage (U). The main control of the Scheduling System is

exercised by a parameter known as N, which at any time is effectively the minimum merit value at which jobs can be run. Thus N varies with time, the system predicting its future values on the basis of past performance. Considered as a graph, N varies over the 24-hour day as the demand for resources varies, with peaks at the busiest times (found in practice to be about 10.15 and 15.00). Experience has shown that average daytime values of N during term time are around 20,000 - 30,000 (peaking at times to 100,000), while night-time values can be in the low thousands.

When scheduling a job, the system consults the current and future predicted values of N and, taking into account the estimated cost of the job and the turnround requested, determines values of M and P by the following procedure. The N value for the time of the requested turnround is considered and the smallest value of P is chosen so that both

$$\frac{S * P * 100,000}{U + W + k * S} > N$$

and

$$P \leq P1$$

(where S, U and W are the shares, usage and outstanding work of the user's project; P1 is the priority limit specified in the job description).

The number $\frac{S * P * 100,000}{U + W + k * S}$ is the merit value M.

The factor k * S is included in the formula to ensure that projects that are new or that have been unused for some time are treated fairly compared to well used projects. The value of k is adjusted from time to time and the current value can be found by using the Phoenix command HELP SCHEDULING K.

The value at September 1976 was 50.

The default priority limit for a job depends on the LIMSTORE and TIME requested and is

255	for a 5 sec and 120K job
127.5	for a 10 sec and 120K job
127.5	for a 5 sec and 240K job
21.25	for a 1 min and 120K job
4.25	for a 5 min and 400K job
1.0625	for a 20 min and 120K job
1	for a 6.375 min or greater and 400K job
1	for a 20 min and 120K or greater job

i.e. P1 = 1275/Ce for most jobs.

These values are such that the addition to the usage U caused by running a job at the default priority limit is independent of the job size, being 1275 resource units for all jobs except the very largest.

Thus the greater the relative values of the current usage (U + W) compared with the share allocation (S), the higher the priority will be needed for a given value of N. However, the lower the value of N at the requested turnround time, the lower P needs to be, thus giving a lower charge. If the requested turnround is immediate (NOW) then the value of N used is the highest in the period lasting twelve times the estimated COMP of the job; if it is a particular time in the future (e.g. TURNROUND BY 1700 or TURN-ROUND OVERNIGHT) then the system chooses the time at which it expects N to be lowest between now and that requested. Thus, the allotted turnround for a job may be less than the requested turnround, but it is important to note that in these cases this is done to provide a lower charge (i.e. lower P). A user may, if he insists, request turnround AT a particular time, but this could give rise to a higher charge if the time he has requested is relatively busy. Note that TURNROUND AT ... OR REJECT should always be used on CHAIN jobs.

If the system is unable to meet the requested turnround (that is, there is no time at which N will be smaller than a merit value calculated at the priority limit (P1) of the job), then this attempt at scheduling is abandoned. As already described the user may explicitly or by default request one or two alternatives (the default is TURNROUND NOW OR OVERNIGHT); these alternatives are tried in order, and if both fail then the job is scheduled for the earliest possible time at the stated priority limit.

The system takes the following particular actions:

Job submission

Ce x P (the estimated charge for the job), is added to the project's W.

The job is scheduled as described above with M and P and the allotted turnround being given.

The job remains in the job queue until it is selected for execution.

Job execution

The job is handed to OS for execution. Note that several jobs will almost always be in execution at the same time.

Job end

The cost of the job (C) is calculated (see Section 2.7.2);

C x P is added to the project's U;

C/(20*S) is added to the project's V (rate of working per share);

Ce x P is subtracted from the project's W.

Each day

Once a day (usually at around 09.00 each morning) each project's U and V are both reduced by about 10%. Thus over a period of time U represents a rate of both resource usage and priority exercised, while V represents the rate of resource usage per allocated share.

Recommended procedures

A new project is initialised with the usage set to zero, and so a newcomer to the system will probably obtain immediate turnround for his smaller jobs even using the default values for TURNROUND and PRIORITY. If the project is then continually underused this happy situation will continue, but usually this is not so and the usage and allotted priority may start to rise. To avoid this, production and non urgent work should be scheduled for evening or OVERNIGHT running by using an explicit TURNROUND statement. Urgent jobs which would only be useful if run immediately can be given the turnround NOW OR REJECT.

Now and then because of an unexpected rise in demand, a job may be given an unacceptable priority or turnround (both figures are given by the SUBMIT command and the latter may also be obtained from the STATUS command). The job may be deleted from the job queue by using the KILLJOB command and resubmitted with new TURNROUND and PRIORITY statements.

It should be noted that the system is designed to permit the user to have a great deal of freedom in his requested turnrounds but it does also aim to limit his overall usage. Thus if an allocation becomes overused it is necessary either to reduce the rate of usage or to have the share allocation reviewed.

Any problems with the Scheduling System should be discussed with a member of User Services Staff (Ext 643 or 641) or with the Superintendent of Computing Services (Ext 636).

Examples

(a)
```
JOB SPQR 1234
PHX
....
```

This job will be given a default turnround of NOW OR OVERNIGHT with a priority limit of 255.

(b)
```
JOB SPQR 1234
LIMSTORE 400K
TIME 10 MINS
JCL
// .....
```

This job will get turnround NOW OR OVERNIGHT with a priority limit of 1. The limit drops to 1 because of the large resource request.

(c)
```
JOB SPQR 1234
TURNROUND NOW OR REJECT
PHX
.....
```

The turnround request is there to force the job to be run immediately or not at all.

(d)
```
JOB SPQR 1234
TURNROUND BY 2000
PHX
.....
```

The job will run at the time giving the lowest priority between now and 20.00. This may be immediately.

(e)
```
JOB SPQR 1234
TURNROUND OVERNIGHT
PHX
.....
```

This would be an appropriate turnround request for a non-urgent job.

(f)
```
JOB SPQR 1234
PRIORITY 1
PHX
.....
```

This job will be run as cheaply as possible.

APPENDIX G

RESTRICTED UTILITIES

There are some IBM programs which are designed to provide facilities for system maintenance, but which are also sometimes invoked by users in the implementation of applications packages or other software. The situation poses problems of safety and compatibility; it is inappropriate for such programs to be generally available to the ordinary user.

The list of 'restricted use' programs may vary from time to time, but the Computing Service will endeavour to give users warning of any change.

Programs at present restricted are:

```
IEHMOVE  user facilities are provided by TRANSFER
IEHPROGM user facilities are provided by XPDS and DELETE
IEHLIST  user facilities are provided by EXAMINE or XPDS
IEHINITT
IEHIOSUP
IFCDIP00
IMAPTFLE
IFCEREP0
IFHSTATR
IEHATLAS
IEHDASDR
```

APPENDIX H

CONTROL CHARACTERS

Sequential datasets consist of a series of "records" which for ordinary textual datasets correspond to lines of text. For example, if the source of a program is stored on disc, each line will occupy one record in the dataset.

Any sequential dataset may be specified as containing "control characters". This means that, as well as the ordinary data in each record, there is one extra character. This is always the first character in each record. If the dataset is sent to a printer, the control character is not printed as part of the line, but instead it is interpreted conventionally as a specification of the spacing required between lines. Hence control characters are sometimes called "carriage control characters". Other unit record output devices may also interpret control characters if they are present. For example, the paper tape output routines use the control character to specify whether a "carriage-return/linefeed" combination is required after each record. Control characters are simply omitted if a dataset containing them is sent to a Phoenix online terminal using the TYPE command.

There are two different sets of control characters, ANSI and machine. The latter are directly related to the control bytes for the printer and punch hardware. The former set are compatible with the control characters used conventionally in FORTRAN, and (for printers) are as follows:-

character	conventional meaning
blank	advance one line before printing
0	advance two lines before printing
-	advance three lines before printing
1	advance to the top of the next page before printing
+	no advance before printing, ie overprint the previous line

A dataset which contains control characters should have one of the letters 'A' (for ANSI) or 'M' (for machine) as part of its record format, and must allow for the control character in its logical record length. For example, a dataset destined for the printer might be set up as

record format (RECFM)	FA
logical record length (LRECL)	133
block size (BLKSIZE)	133

Most of the problems concerning control characters occur when a dataset containing them is stored on disc or magnetic tape and is subsequently processed by program. The program can discover whether the dataset it is reading contains control characters or not, but it is entirely free to process them as it likes. Different conventions, all of which have their pros and cons, have unfortunately been adopted by different programs.

The utility programs FILE and EDIT, for example, differ in the way they behave when one of the input and output datasets contains control characters and the other does not. This situation can easily arise in Phoenix jobs and in the Phoenix online system because the default record format for a new output dataset is VB, and this is independent of the input format.

FILE copies all the characters in each record from input to output, whatever the format of either. This has the merit that no characters are ever lost. If a dataset which is labelled as containing control characters is FILEd to one without, the control character will become the first data character in each record, and will appear if the dataset is printed or typed. If, on the other hand, a dataset without control characters is FILEd to one that is labelled as having them, the first data character in each record will become the control character, and will appear to have "vanished" if the dataset is typed or printed, apart from possible obscure carriage control effects.

EDIT takes note of the control character specifications of its input and output, and it removes control characters when going from a dataset that has them to one that does not, and adds a default control character in the opposite situation. This convention causes control characters to be lost in the one case; on the other hand it preserves the data characters as data characters. Control characters in records are not included in EDIT's context editing. Special commands are provided for setting the control character in the current line.

Most compilers and utilities that originated elsewhere do not allow control characters in their input, though they frequently generate them in their output. Compilers and utilities that originated in Cambridge (eg BCPL, ALGOL68C, ML/I, XPDS, PATCH) are generally more flexible in their input requirements. As a general rule, though, it is safer to avoid using datasets with control characters unless some special feature, for example underlining, is needed.

APPENDIX I

Using the DEC GT40

The DEC GT40 is provided in the Reception Area for any user who wishes to look at his plotted output before sending it to the plotter. The GT40 may also be used as a Phoenix console when it is not required for graphical work. It has several facilities which are not normally available on a Phoenix console, including a stack of four buffers. A complete description of the DEC GT40 is given in the Graphics Manual, but a list of features is given here.

GT40 Function Keys

C.R.	Send line and make vulnerable
L.F.	Send line and leave as current
Rubout	Delete character left of cursor
Tab	Tab
Alt	Break
Home	Clear input area
Up arrow	Pop from stack
Down arrow	Push onto stack
Left arrow	Cursor left
Right arrow	Cursor right

GT40 Control Characters

Control characters are entered by pressing the CNTRL key and a letter simultaneously.

A*	Reset ASCII tabs
B+	Enter bootstrap
C*	Display commands
D	Delete graticule
E	Erase screen
F	Roll forward (up)
G	Display graticule
H	Cursor left
I	Tab
J	L.F.
K	Push onto stack
L	Not used
M	C.R.
N	Not used
O	Not used
P	Not used
Q	Not used
R	Roll backwards (down)
S*	Set tabs
T*	Display tabs

\+ This facility is for specialist use only.
* Not yet available.

U	Default input to upper case
V	Not used
W	Start interval signal
X	Cursor right
Y	Default input to lower case
Z	Pop from stack
Λ	Repeat

NOTES

(a) A line is vulnerable, i.e. may be overwritten, when it appears at half intensity.

(b) Line feed does not send vulnerable lines.

(c) To pop from stack is to make current.

(d) To push onto stack is to take item from bottom of stack and make vulnerable.

Part II

Specifications of Commands and Procedures

This section of the manual contains specifications of the Phoenix commands, procedures and switches and of the JCL catalogued procedures available on the 370/165. The heading for each topic gives an indication of whether the subject concerns the JCL user, the Phoenix user, or both; and for all topics the description is intended to describe how to use a facility by giving both examples and syntax of the various commands and procedures. For the Cambridge developed software this is followed by a complete specification of the facility, while references are given to other documentation for further details of non-Cambridge software. Users are advised also to consult the manual 'Summary of Applications Software', obtainable from the Computing Service Bookshop, if they cannot find the information they require about some language or package in this manual. The following topics may be found in Part II in alphabetical order:

%	FALSE	OLDABEND	SPOOL command
ABANDON	FAST	ONLINE	SPSS
ABEND	FD	PARS	STATUS
ALGOL (W)	FILE	PASCAL	SUBMIT
ALGOL 60	FILEPDS	PASSWORD	SUGGEST
ALGOL 68	FINISH	PATCH	SWITCH
ASSEMBLER	FORM	PDSRECOV	TAPECOPY
BADFILE	FORTRAN	PDSSCAN	TAPESCAN
BADP	HELP	PLI	TAPESTAT
BADPARM	HELPK	PRINT	TERMINAL
BADSYNT	HP	PROJECT	TESTTERM
BCPL	IBMCMPRS	PROTECT	TLS
C	IBMCOPY	PUNCH	TLSCOPY
CHAR	IF	QUIT	TLSINIT
CLEAR	IMAGE	RC	TLSRECOV
COBOL	INFO	RELATION	TLSSCAN
COLLECT	INPUT	RELEASE	TO
COMMENT	JUMP	RENAME	TRANS
COMPARE	KILL	REQUEST	TRANSFER
COMPRESS	LABEL	REROUTE	TRUE
CONDENSE	LIBRARY	RETURN	TYPE
COPY	LINKAGE EDITOR	RUBOUT	UNLESS
CREATION	LIST	RUN	USER SWITCHES
CROSSREF	LOCAL	SAVE	VARIABLES
CURRENT	LOGON	SCRATCH	VIEW
DELETE	MESSAGE	SEND	WAIT
DISPLAY	MKINIT	SET	XLABEL
EDIT	MKLIB	SETUP	XMODULE
EDSORT	MLI	SKIP	XPDS
ERRJUMP	NO	SNOBOL	XTAPE
ERROR	NOTIFY	SORT-MERGE	YES
EXAMINE	NULL	SPACE	

Copyright

Certain items of software described in this manual are designated proprietary. The attention of users of this software is drawn to Rule 5 of the Rules made by the Computer Syndicate (Reporter No.4870 31st July 1974), and to the following notice:

Reference to proprietary software may only be made through the procedures or library call mechanisms provided for the purpose, or by such other methods approved by the University Computing Service. The files containing the information may not be copied or printed, either in whole or in part, nor may a core image be dumped or printed.

Procedures and Symbolic Parameters

Both the Phoenix procedures and the JCL catalogued procedures make extensive use of symbolic parameters which are used to specify datasets or to change options. The names of the symbolic parameters have been designed to be mnemonic and consistent between procedures, so that after using a few of the procedures, it is hoped that users will not find it necessary to consult the documentation on symbolic parameters unless they are trying to change an obscure option.

JCL catalogued procedure names for the utilities are in general just the same as the utility name (e.g. FILE, LIST, LISTPDS, CONDENSE) because they each only require a single procedure. The exception is SORT/MERGE which has a choice of four procedures. Under Phoenix the utilities are generally invoked by macro commands rather than by procedures, the exceptions again being SORT/MERGE.

Two very common symbolic parameters used with the utility procedures are NAME and OPT. NAME is the parameter normally used to refer to any single dataset that the utility is processing (e.g. filing, printing, listing etc.) while OPT is used to give small numbers of options to the utility. Most of the other parameters are mnemonic and are terminated by an integer which corresponds to the step number within the procedure. Thus the majority of the utility procedures contain a parameter REG1 which gives the region size for the first step, and MIN1 and SEC1 which specify the time limit for that step.

The names for the language procedures can be of two forms. If the procedure invokes a compile-and-go processor, the procedure name is normally simply that of the language processor (e.g. WATFIV and SNOBOL), while if it is a multistep processor, the name consists of a mnemonic prefix for the language processor (e.g. FORTX, FTG1, PLICK, PLIX etc.) followed by a combination, in order, of character C for compile step, L for linkage edit step and G for the go step. Thus the Fortran G1 compile, link edit and run procedures is called FTG1CLG both in JCL and Phoenix.

The stepnames in the JCL procedures are similarly mnemonic. The linkage edit step is always given the name LKED and the run step the name GO. The compile steps are named FORT for the IBM Fortran compilers, PLI for the PL/I checkout and optimiser compilers and so on.

Symbolic parameter names for both JCL and Phoenix language procedures are generally terminated with the letter C, L or G defining to which step the parameters refer. The prefix describes their purposes. For example, the listing option in a compile step is commonly changed using the parameter LISTC, and region size and time by REGC, MINC and SECC. Similarly in the link edit steps there are parameters LISTL, REGL, MINL and SECL; and in the GO step, REGG, MING and SECG.

There is one important difference between the Phoenix and the JCL parameters in that the program is normally given to a JCL procedure by using a dd statement with the ddname SYSIN, while in the Phoenix procedures it is given either with a symbolic parameter PROGRAM or a ddname PROGRAM

For a full list of the parameters for any JCL procedure, the dataset INFO.JCL PARMS(procedure-name) should be printed, or the copy of this dataset in the User Area may be consulted. For Phoenix procedures the command 'HELP PROC procedure-name' should be used.

Users who are unfamiliar with the use of symbolic parameters are advised to read the relevant sections in the manual 'Basic Use of the Cambridge 370/165'.

Phoenix Macro Commands

Most of the commands used to invoke the utilities in Phoenix are Phoenix Macro Commands, the construction of which is described in Section 4.7. It will be seen in that Section that the macro-name is followed by keyword-value-pairs, which are used to define datasets and to supply options, and that both the keyword-value-pairs and the keywords may be mandatory or optional. Each macro command description contains a syntax section which lists the keywords and the datatype of the values, and also, if appropriate, any default values built into the commands. The majority of the syntax sections need to be studied in conjunction with the footnotes given at the bottom of the page. Preceding the syntax description is a set of examples, which should suffice as a guide to the more common uses of each command, so that the syntax section need only be consulted for more esoteric uses or for the complete command description.

SYSTEM VARIABLES

System variable names always begin with %. They should be distinguished from system commands which can be prefixed with % to distinguish them from user commands with the same name. The system variables available are:

%ABEND	This holds the abend code of the last abending command in system format. %ABEND = %SCC * 2**12 + %UCC.
%CCPU	This holds the cpu time used so far in the job or console session in centiseconds.
%CDISCIO	This holds the disc I/O time used so far in the job or session in centiseconds.
%CHARGE	This holds the charge for the job or session so far in units of (priority x resource units).
%CHECKED	This states whether the job or session is in password checked state.
%CPU	This holds the cpu time use so far by the job or session.
%CRU	This is set by the system to the number of resource units used so far by the job or console session in centiunits.
%CTAPEIO	This holds the magnetic tape time used by the job in centiseconds.
%DATE	This holds the Gregorian date.
%DAY	This is set by the system to the day of the month.
%DAYNAME	This is set by the system to the day name.
%HOUR	This is set to the hour of the day using a 24 hour clock.
%IBMDATE	This holds the date in IBM format.
%JNUMBER	This is set by the system to the job number.
%LEVEL	This is set by the system to the level number of the current command sequence.
%MIN	This is set to the minute of the current hour.
%MONTH	This holds the month of the year (JANUARY etc.).
%PRIORITY	This is set by the system to the priority of the job or console session.
%PROJECT	This is set to the project number under which the console session or job is running.

%
2

%PWARNLEVEL This may be set by the user to a number above which a change in priority in a console session is to be notified. The default value is 100

%QLEVEL This may be set to an integer. If the QUIET switch is unset and %QLEVEL is set to n, then command reflections and information messages are suppressed in command-sequence levels greater than n. If the QUIET switch is set, %QLEVEL has no effect. If %QLEVEL has not been explicitly set since the beginning of a job or console session, then setting the QUIET switch will cause %QLEVEL to be set to 0 and in setting it will cause %QLEVEL to be set to 15. %QLEVEL is initially set to 15.

%RC This holds the return code of the last command.

%RCLEVEL This may be set to an integer. If %RCLEVEL is set to n, then if any command or procedure produces a return code less than or equal to n the error state will be suppressed, i.e. the RC and ERROR flags will not be set and the ERRCODE message will not be produced, although %RC will contain the return code. %RCLEVEL is a global variable which is initially set to zero. It is restorable with SW RESTORE.

%RDYTEXT This may be set by the user to change the text which is produced when Phoenix returns control to a console. The text given to %RDYTEXT may contain occurrences of a symbolic parameter. To unset %RDYTEXT either SET %RDYTEXT or SET %RDYTEXT '' may be used.

%REGION This is set to the region size of the job or console session.

%RESOURCE This holds the resource units so far in the job or console session.

%SCC This holds the system completion code of the last abending command.

%SEC This is set to the seconds of the current minute.

%SET This is the text from the last SET NAME command. %SET is initially set to the logged on userid or the userid under which the job is being run.

%TAPEIO This is set to the tape time used so far in the job.

%TIME This holds the current time of day.

%UCC This holds the user abend completion code of the last abending command.

%USER This is set by the system to the logged on userid or the user identifier under which the job is being run.

%WIDTH This holds the terminal width as set by the SET TERMINAL WIDTH command.

%
3

%YEAR This holds the year number (eg. 1978).

All the above may be tested in IF commands and may be used in symbolic parameter occurrences.

PHOENIX EXAMPLES

(a) IF ABEND SKIP ERROR
....
*ERROR
** ABEND is <%ABEND>

This tests the system flag ABEND to see if an error has occurred. If so, a jump is made to the error handling routine and the value of the abend produced in the comment.

(b) SET %QLEVEL=1
FTG1CLG PROGRAM=.PROG

This suppresses command reflection from within the FTG1CLG procedure.

(c) SET %RCLEVEL=4

Any subsequent commands will only set the RC and ERROR switches if they produce a return code greater than 4.

(d) IF %RC GT 4 SKIP TWADDLE
....
*TWADDLE

This tests the system variable %RC to see if a previous command has produced a return code greater than 4. If %RC is less than or equal to 4, the command stream continues to be obeyed, otherwise control is transferred to the label TWADDLE and execution continues.

ABANDON

ABANDON is a Phoenix control command which returns control to the console when used in a console session, or terminates the job step when used in a job. It may be followed by an unsigned integer which is used to set a return code. A is synonymous with ABANDON. See Section 4.10.6 for further details.

EXAMPLES

(a) IF ERROR ABANDON

Control returns to the console or the job step is terminated if the ERROR flag is set.

(b) ABANDON 999

Control returns to the console or the job step is terminated, and a return code of 999 is set.

ABEND

ABEND is a Phoenix flag which may be tested by the user. It is set if the last command terminated with an ABEND, and is unset when the next non-control command is obeyed. See Section 4.10.2 for further details.

EXAMPLES

(a) IF ABEND ABANDON

If an ABEND occurred in the last non-control command the ABEND flag will be set and so control is transferred to the console or the job step is terminated. Otherwise the command stream continues to be obeyed.

(b) UNLESS ABEND SEND ABC1 RUN SUCCESSFUL

If the ABEND flag is set the command stream continues to be obeyed, otherwise a message 'RUN SUCCESSFUL' is sent to user ABC1 and the command stream continues to be obeyed with the ABEND flag cleared.

ALGOL W

ALGOL W is a high level programming language developed from ALGOL 60 and intended mainly for student teaching. It is unsuitable for large programs. There are two versions of the compiler, the preferred one from Stanford and one from Newcastle each of which may be used via JCL catalogued or Phoenix procedures. Details of the language and the compilers may be found in the manual 'ALGOLW Language Specification and Programmers Guide'.

PROCEDURES

The following JCL catalogued procedures and Phoenix procedures are available:

The standard (Stanford) version:
ALGOLX - for the compile-and-go batching system
ALGWC
ALGWCL
ALGWCLG

The Newcastle version may be used through these procedures, but with the following symbolic parameters set:

for ALGOLX, ALGWC, ALGWCL, ALGWCLG - SYSTEM=NC

Note that the Phoenix procedures may not be used in a Phoenix console session.

PHOENIX EXAMPLES

(a) ALGOLX PROGRAM=.PROG

compiles and runs the ALGOL W program contained in the dataset .PROG.

(b) ALGOLX (PROGRAM = %H$) SYSEM=NC

```
 END.
$
```

compiles and runs a program contained in the %H dataset using the Newcastle compiler.

ALGOL W
2

(c) ALGWCLG PROGRAM=.PROG LIBRARY=.LIB

compiles, link edits and runs a program in .PROG picking up procedures by automatic library scan from the dataset .LIB

Further examples may be found in Example Sheet 15.

PHOENIX PARAMETERS AND DDNAMES

The program or procedure must be supplied to the procedures using either the symbolic parameter or the ddname PROGRAM. Apart from PROGRAM all the symbolic parameters in ALGOLX finish with G, while in the other procedures the usual convention of termination with C, L or G is followed. In the final step of the ALGWCLG procedure the primary data input is given by either the symbolic parameter or the ddname INPUT. There is also for this step a parameter TIMEG (default value 4) which controls the time in seconds given to the command by the ALGOLW system.

A complete list of parameters for each procedure may be obtained by using the command 'HELP PROC procedure-name'.

JCL EXAMPLES

(a)
```
// EXEC ALGOLX
//SYSIN DD DSN=SPQR.PROG,DISP=SHR
```

This is a compile-and-go execution of the program contained in the dataset SPQR.PROG.

(b)
```
// EXEC ALGOLX,SYSTEM=NC
//SYSIN DD *

  END.
//*
```

This compile-and-go execution uses an instream dataset and the Newcastle compiler.

(c)
```
// EXEC ALGWCLG
//ALGW.SYSIN DD DSN=SPQR.PROG,DISP=SHR
//LKED.SYSLIB DD
//       DD DSN=SPQR.LIB,DIPS=SHR
```

This is a compilation, linkage editing and running of a program from SPQR.PROG with external references being resolved by the automatic library scan of SPQR.LIB.

Further examples may be found in Example Sheet 15.

ALGOL W
3

JCL PARAMETERS AND DDNAMES

The program must be in a dataset defined by a dd statement with name SYSIN. A full list of symbolic parameters may be obtained from the partitioned dataset INFO.JCL.PARMS using the member name which corresponds to the procedure name. Note the comment about TIMEG in the above Phoenix section which is also applicable to the JCL parameter.

USE OF FORTRAN SUBROUTINES WITH ALGOL W

Any Fortran routines called must be suitably declared as FORTRAN and the appropriate library scanned by concatenation with the LKED.SYSLIB DD statement or, with Phoenix, by use of LIBRARY=. ALGOLW does not initialise the Fortran error package and, therefore, difficulty may be encountered if the program fails within the routine. S.FORTLIB should be scanned for mathematical routines; no advantage is gained by using SYS1.FORTLIB as far as error messages are concerned, but if Fortran I/O is used then SYS1.FORTLIB is essential (S.FORTLIB does not contain the I/O package). The hard fail option in NAG routines cannot therefore be used. The procedure described for the use of the soft fail option with PL/I should be followed. If an error occurs, advice should be sought from the programming adviser. Further details may be obtained from the manual 'CAMLIB Subroutine Library Specifications'.

FORTRAN SUBROUTINE EXAMPLES

(a)
```
// EXEC ALGWCLG
//ALGW.SYSIN  DD  *
  BEGIN
    LONG REAL X,T;  INTEGER IFAIL;
    LONG REAL PROCEDURE S07AAF(LONG REAL VALUE X;
                               INTEGER VALUE IFAIL);
       FORTRAN  "S07AAF" ;
    ....
       T:=S07AAF(X,IFAIL);
    ....
    END.
//*
//LKED.SYSLIB  DD
//             DD DSN=SYS2.NAGLIB,DISP=SHR
//             DD DSN=S.FORTLIB,DISP=SHR
```

(b)
```
ALGWCLG (PROGRAM %H!) LIBRARY=SYS2.NAGLIB+S.FORTLIB
        ....
     program as above
        ....
!
```

NOTES

Further information may be obtained from the manual 'ALGOLW Language Specification and Programmers Guide'.

ALGOL 60

ALGOL 60 is a high level programming language available in the two versions IBM ALGOL F and Delft ALGOL. The latter is the recommended compiler (even by IBM) as it is smaller, faster, has better error diagnostics and may be used as a load and go system, and as ALGOL F is no longer maintained by IBM. Both compilers may be accessed via JCL catalogued procedures or via Phoenix procedures.

Detailed information about the ALGOL F compiler may be found in the IBM manuals 'OS ALGOL Language Reference Manual' (GC28-6615) and 'OS ALGOL Programmer's Guide' (GC33-4000); and about the Delft ALGOL compiler in 'Delft ALGOL 60 Compiler Programmer's Guide' (SB10-6045).

PROCEDURES

The following JCL catalogued procedures and Phoenix procedures are available:

For ALGOL F : ALGOFC, ALGOFCL, ALGOFCLG, ALGOFLG.

For Delft ALGOL : ALGOLC, ALGOLCL, ALGOLCLG, ALGOLLG,
ALGOL (for the one-step compiler).

Note that the Phoenix procedures are not available in a console session.

PHOENIX EXAMPLES

(a)
```
ALGOL  (PROGRAM=%H#)
     'PROCEDURE'
     .....

     #
```

The example uses the Delft ALGOL compiler to provide load-and-go execution of an ALGOL program given in a %H dataset.

(b)
```
ALGOL PROGRAM=.PROG CPU=1000 PGS=20 DATAFILE=.DATA
```

This is load-and-go execution of the program contained in .PROG changing the compiler time limit to 10 secs (1000 centisecs) and increasing the page limit to 20 pages. Data is provided by using the symbolic parameter DATAFILE to define the dataset .DATA.

(c)
```
ALGOLCLG  (PROGRAM=%H!)  LISTC=5  OPTC=1  LIBRARY=.MYLIB
     .....
     !
```

This is the Delft ALGOL compiler used as a multistep compiler. The main procedure is given as a %H dataset; a source listing is requested using symbolic parameter LISTC and optimisation using OPTC, and the library of pre-compiled procedures .MYLIB scanned by using the symbolic parameter LIBRARY.

ALGOL 60
2

(d) ALGOLCLG (ALGLDD02=*A ALGLDD03=.RES/N) PROGRAM=.PROG PARMG=TRACE

The Delft ALGOL compiler is again used to run the program contained in .PROG, with two extra execution time datasets defined in the environment with ddnames ALGLDD02 (set up to be a printer dataset) and ALGLDD03 (set up to define a new disc dataset .RES).

Further examples may be found in Example Sheet 14.

PHOENIX PARAMETERS AND DDNAMES

For all compilation procedures the program or procedure may be given using either the symbolic parameter or the ddname PROGRAM.

The load-and-go Delft ALGOL procedure ALGOL has some symbolic parameters that have no equivalents in the other procedures, for example CPU (default 500) and PGS (default 10) are used to control the total cpu time in centiseconds and the total output in pages. The primary output for listings and error messages may be changed from the default (a printer dataset) by using the symbolic parameter TO; and data for dataset number 0 is defined by using either the symbolic parameter or the ddname DATAFILE. Note that the load-and-go compiler needs a region size of at least 170K

The remainder of the Delft ALGOL procedures have the following useful symbolic parameters. LISTC (default NOSOURCE) controls the compiler listing option for the source program, while OPTC, which has a default of 0, meaning no optimisation, can be used to switch on optimisation by giving it the value 1. Load module libraries to be scanned may be given as the value of the LIBRARY parameter Data for execution time to be read as dataset number 0 may be given using either the symbolic parameter or the ddname DATA , and the execution time option for tracing may be switched on by using the symbolic parameter PARMG.

Details of the ALGOL F parameters and further details on the Delft ALGOL parameters for each procedure may be obtained by using the command 'HELP PROC procedure-name'.

JCL EXAMPLES

(a)
```
// EXEC ALGOL
//PROGRAM  DD  *
   'PROCEDURE'

//*
```

uses the Delft ALGOL compiler to provide load-and-go execution of an ALGOL program given as an instream dataset.

ALGOL 60
3

(b)
```
// EXEC ALGOL,CPU=1000,PGS=20
//PROGRAM  DD  DSN=SPQR PROG,DISP=SHR
//DATA  DD  DSN=SPQR DATA DISP=SHR
```

This is load-and-go execution of the program contained in SPQR.PROG with data for dataset unit number 0 provided on the dd statement with the ddname DATA The cpu time limit is increased to 1000 centiseconds and the page limit to 20 pages.

(c)
```
// EXEC ALGOLCLG,LISTC=S,OPTC=1
//ALGOL.SYSIN  DD  *

//*
//LKED SYSLIB  DD
//            DD  DSN=SPQR MYLIB,DISP=SHR
```

The Delft ALGOL compiler is used as a multistep compiler to compile an instream program The symbolic parameters LISTC and OPTC are given values to produce a source listing and optimisation. Pre-compiled procedures are picked up by providing a concatenated overriding SYSLIB dd statement.

JCL PARAMETERS AND DDNAMES

The program must be supplied with a ddname PROGRAM in the ALGOL procedure and with SYSIN in the others

The load-and-go Delft ALGOL procedure ALGOL has some symbolic parameters that have no equivalents in the other procedures, for example CPU (default 500) and PGS (default 10) are used to control the total cpu time in centiseconds and the total output in pages. The primary output for listings and error messages may be changed from the default (a printer dataset) by using the symbolic parameter TO or by an overriding dd statement with ddname ALGLDD01; and data for dataset number 0 is defined by using either the symbolic parameter DATAFILE or the ddname DATA. Note that the load-and-go compiler needs a region size of at least 170K

The remainder of the Delft ALGOL procedures have the following useful symbolic parameters LISTC (default NOSOURCE) controls the compiler listing option for the source program while OPTC, which has a default of 0 meaning no optimisation, can be used to switch on optimisation by giving it the value 1 Data for execution time to be read as dataset number 0 must be given using a dd statement with name SYSIN; and the execution time option for tracing may be switched on by using the symbolic parameter PARMG

Details of the ALGOL F parameters and further details of the Delft ALGOL parameters may be obtained by listing the dataset
INFO.JCL PARMS(procedure-name)

ALGOL68C

ALGOL68C is a versatile programming language, identical in many respects to ALGOL 68 as defined by the 'Revised Report on ALGOL 68'. The language ALGOL68C and the compiler were developed as a research project at the University of Cambridge Computer Laboratory. The implementation is not yet complete, but it is available for use. Until the implementation is complete, it will not be suitable for the novice programmer.

REFERENCES

Example Sheet 30

'A Guide to the Use of ALGOL68C on the 370/165 at Cambridge' - by M. F. Challis

'ALGOL68C Reference Manual' - by S. R. Bourne, A. D. Birrell and I. Walker

(none available at the time of printing)

PROCEDURES

The following JCL and Phoenix procedures are available:-

A68C
A68CL
A68CLG
A68LG
A68XREF

The Phoenix procedures cannot be used in a Phoenix console session.

Note that the 'compile' stage for ALGOL68C consists of two steps; the first compiles ALGOL68C source to an intermediate language called ZCODE, and the second translates ZCODE to a standard IBM object module.

PHOENIX EXAMPLES

(a) A68CLG PROGRAM=.PROG DATA=.INPUT

compiles and executes the ALGOL68C program held in the dataset .PROG with data from dataset .INPUT available on the standard input file.

(b) A68CL PROGRAM=.SEGA NAMEL=.MODS(A) ENVOUT=.ENVS(SEGA)/N

compiles the 'root' segment of a multisegment program, storing the resulting environment in .ENVS(SEGA) and the compiled load module in .MODS(A).

ALGOL68C
2

(c) SET CHAR END #
A68CL (ENVS= ENVS) PROGRAM= SEGZ NAMEL=.MODS(Z) #
PARMC='/USING XYZ FROM "ENVS(SEGY)"'

compiles a 'leaf segment of a multisegment program; the environment containing the definitions necessary for the correct compilation of the segment is set up on ddname ENVS

(d) SET CHAR END #
A68CL(ENVS= ENVS) PROGRAM= SEGN NAMEL=.MODS(N) #
ENVOUT= ENVS(SEGN) #
PARMC='/USING PQR FROM "ENVS(SEGM)"'

compiles an 'internal' segment of a multisegment program. ENVS makes available existing environment information and ENVOUT specifies where the new environment is to be stored.

(e) A68IG (LKIN=%H# LIB= MODS) DATA=.IN7
INCLUDE LIB(A,B C)
#

link-edits and executes a three segment program composed of modules A, B and C; these are supposed to have been previously compiled in ways analogous to those shown above at (b), (c) and (d).

PHOENIX PARAMETERS AND DDNAMES

REGC, REGT, REGL REGG and SECC. SECT, SECL, SECG control the region sizes and times for the compilation, translation, link-editing and execution steps respectively in the usual way - note the suffix 'T' for the translate step

The source program may be set up on ddname PROGRAM or preserved in a dataset whose name is given as the value of the parameter PROGRAM. Program headings may appear (preceded by '/') as the value of parameter PARMC

The DATA parameter specifies data for the standard input and the parameter PRINTG defines the standard output. Characters following '/' on PARMG define the contents of the book on the parameter channel.

ENVOUT specifies the destination of any environment produced during compilation, and NAMEL specifies the destination for a compiled program in load module format

A complete list of Phoenix parameters may be obtained for each procedure by using the command 'HELP PROC procedure-name'.

ALGOL68C 3

JCL EXAMPLES

(a)
```
// EXEC A68CLG
//A68.SYSIN DD  DSN=MFC9.PROG,DISP=SHR
//GO.SYSIN  DD  *
  17  35
//*
```

compiles and executes the ALGOL68C program held in the file MFC9.PROG with the instream data made available on the standard input file.

(b)
```
// EXEC A68CL
//A68.ENVOUT  DD  DSN=MFC9.ENVT(SEGA),UNIT=DISC,DISP=(,CATLG),
//         SPACE=(TRK,(2,2,2)),DCB=VB
//A68.SYSIN DD  DSN=MFC9.SEGA,DISP=SHR
//LKED.SYSLMOD  DD  DSN=MFC9.MODS(A),UNIT=DISC,DISP=(,CATLG),
//    SPACE=(TRK,(2,2,2))
```

compiles the 'root' segment of a multisegment program. The resulting environment is stored as the first member SEGA of a newly created pds MFC9.ENVT, and the compiled load module is stored as the first member A of a newly created library MFC9.MODS.

(c)
```
// EXEC A68CL,NAMEL='MFC9.MODS(Z)',DISPL=OLD,
//  PARMC='/USING XYZ FROM "ENVS(SEGY)"'
//A68.SYSIN  DD  DSN=MFC9.SEGZ,DISP=SHR
//A68.ENVS  DD  DSN=MFC9.ENVT,DISP=SHR
```

compiles a 'leaf' segment of a multisegment program; the environment containing the definitions necessary for the correct compilation of the segment is set up on ddname ENVS.

(d)
```
//  EXEC A68CL,NAMEL='MFC9.MODS(N)',DISPL=OLD,
//  ENVOUT='MFC9.ENVT(SEGN)',
//  PARMC='/USING PQR FROM "ENVS(SEGM)"'
//A68.SYSIN DD  DSN=MFC9.SEGN,DISP=SHR
//A68.ENVS DD   DSN=MFC9.ENVT,DISP=SHR
```

compiles an 'internal' segment of a multisegment program. ENVS makes available existing environment information and ENVOUT specifies where the new environment is to be stored.

(e)
```
//  EXEC A68LG
//LKED.LIB  DD  DSN=MFC9.MODS,DISP=SHR
//LKED.SYSIN DD  *
    INCLUDE LIB(A,B,C)
//*
//GO.SYSIN  DD  DSN=MFC9.IN7,DISP=SHR
```

link edits and executes a three segment program composed of modules A, B and C; these are supposed to have been previously compiled in ways analogous to those shown above in (b), (c) and (d).

ALGOL68C
4

JCL PARAMETERS AND DDNAMES

Step names are A68, TRAN, LKED and GO.

REGC, -T, -L, -G and MINC, -T, -L, -G, SECC, -T, -L, -G control the region sizes and times for the compilation, translation, link-editing and execution steps respectively in the usual way - note the suffix 'T' for the translate step.

The source program must be set up on an additional dd statement of the form

```
//A68.SYSIN  DD etc
```

A program heading may appear (preceded by '/') as the value of parameter PARMC.

Data for the standard input must be defined by a dd statement of the form

```
//GO.SYSIN  DD etc
```

but the standard output dataset may be given as the value of the parameter PRINTG. Characters following '/' on PARMG define the contents of the 'book' on the parameter channel.

ENVOUT specifies the destination of any environment produced during compilation, and NAMEL, DISPL together specify the destination for a compiled program in load module format in the usual way.

ASSEMBLER

There are four assemblers available for the IBM OS Assembly Language, three of which are accessed via Phoenix procedures or JCL catalogued procedures. These three are Assembler F, IBM's basic language translator; Assembler H, an IBM Program Product, and Assembler G which was developed by the University of Waterloo. This last is the recommended assembler for use in most circumstances, having superior facilities to F and being faster ard smaller than H, although H should be used for large programs with large numbers of macros.

The fourth assembler is a one pass system called SPASM, (developed by Stanford University) which is suitable mainly for course use.

REFERENCES

'OS Assembler Language'	(GC28-6514)
'OS Assembler F Programmer's Guide'	(GC28-3756)
'OS Assembler G User's Guide'	
'OS Assembler H Language specification'	(GC28-3771)
'OS Assembler H Programmer's Guide'	(SC26-3759)
'OS Assembler H General Information Manual'	(GC28-3758)
'SPASM User's Guide'	
'System/370 Principles of Operation'	(GA22-7000)
'Cambridge Aids to Assembler Progammers'	

PROCEDURES

The following JCL catalogued procedures and Phoenix procedures are available:-

Assembler F :	ASMFC,ASMFCL,ASMFCLG,ASMFLG,ASMFG
G :	ASMGC,ASMGCL,ASMGCLG,ASMGLG
H :	ASMHC,ASMHCL,ASMHCLG,ASMHLG

Note that the Phoenix procedures cannot be used in a Phoenix console session.

PHOENIX EXAMPLES

(a) ASMGCL PROGRAM=.SOURCE MACLIB=.PRIMAC NAMEL=.LOADLIB(CXYV)

A control section is assembled from .SOURCE using a private macro library .PRIMAC defined by the symbolic parameter MACLIB, and is then link edited with the output load module being stored in the library .LOADLIB.

(b) ASMGCLG (PROGRAM=%H! OUT=*A) LIBRARY=.LOADLIB

!

A source module defined in a %H dataset is assembled and link edited with external references being resolved from the library .LOADLIB. The resulting load module is run using a ddname of OUT for output to the printer.

Further examples may be found in Example Sheet 11.

PHOENIX PARAMETERS AND DDNAMES

The source module must be given using either the symbolic parameter or the ddname PROGRAM, while macro libraries in addition to SYS1.MACLIB can be defined by the symbolic parameter MACLIB. Further useful parameters are LISTC used to control listing option, RENTC for the re-enterability option and BATCHC (G and H only) for batch assemblies.

The values these and other parameters may take vary slightly between the assemblers and users are urged to use the command 'HELP PROC procedure-name' to obtain complete details of the symbolic parameters and their defaults for each procedure.

JCL EXAMPLES

(a)
```
// EXEC ASMGCL
//ASM.SYSLIB DD
//      DD DSN=SPQR.PRIMAC,DISP=SHR
//ASM SYSIN DD DSN=SPQR SOURCE,DISP=SHR
//LKED SYSLMOD DD DISP=SHR,DSN=SPQR.LOADLIB(CXYV)
```

A control section is assembled from the disc dataset SPQR.SOURCE using the private macro library SPQR.PRIMAC, which is defined on the overriding and concatenated dd statement with ddname SYSLIB The output load module from the link-edit step is stored in the private linkage library SPQR.LOADLIB.

(b)
```
//  EXEC ASMGCLG
//ASM.SYSIN DD *

.     .  .

//*
//LKED.SYSLIB DD
//             DD  DSN=SPQR.LOADLIB,DISP=SHR
//GO OUT  DD  SYSOUT=A
```

The source module is defined as an instream dataset and unresolved external references are obtained from the library SPQR.LOADLIB which is given on an overriding and concatenated SYSLIB dd statement. A dd statement with ddname OUT is added for go-step output to printer.

ASSEMBLER
3

JCL PARAMETERS AND DDNAMES

Full details of available parameters for each procedure may be obtained from the dataset INFO.JCL.PARMS(procedure-name).

SVC'S AND MACROS

SVC's (except those described in 'Cambridge Aids to Assembler Programmers') may only be used via assembler macro instructions as they are defined only at the macro level.

The following Cambridge macros are described in 'Cambridge Aids to Assembler Programmers'.

INFO	gives date and time in various formats.
PM	gives post mortem output of core, subpools or load modules.

Also in the same publication are details of the alternatives to SYSUDUMP called CAMDUMP and CAMXDUMP which give easy to interpret ABEND dumps.

Generally, the only system macros that should be used are the following:-

ABEND	FREEDBUF	SAVE
ATTACH	FREEMAIN	SEGLD
BLDL	FREEPOOL	SEGWT
BSP	GET	SETL
BUILD	GETBUF	SNAP
BUILDRCD	GETMAIN	SPIE
CALL	GETPOOL	STAE
CHECK	LINK	STIMER
CLOSE	LOAD	STOW
DCB	NOTE	SYNADAF
DCBD	OPEN (not TYPE=J)	SYNADRLS
DELETE	POINT	TIME
DETACH	PUT	TRUNC
DXR	PUTX	TTIMER
EXTRACT	READ	WRITE
FEOV	RELEX	WTO (only routecode 11)
FIND	RELSE	XCTL
FREEBUF	RETURN	

The majority of system macros, as specified in the System Programmers Guide, are forbidden to users. They are unavailable either because they cannot be guaranteed to remain unchanged, or because their use could be in some way dangerous to the system itself, to the user or to users in general, or because the macro is not relevant to the Cambridge configuration.

However, it is sometimes expedient to relax these rules for particular users who have special requirements. The following list indicates certain macros which are thought to be 'soft' in that they probably do

ASSEMBLER
4

not provide security loopholes into the system, nor are likely to cause system crashes if used correctly. Experienced users may use these macros, but are urged to note the following.

(a) The macros should be used only if more conventional techniques are really inadequate.

(b) If they are in any doubt, users should first seek the advice of Computing Service staff.

(c) The specification of these macros is not subject to the same guarantee against change as for ordinary macros.

If it is found necessary to consider changing the specification of any of these macros, the Computing Service will endeavour to give adequate notice. Users can best safeguard themselves in this respect by lodging with the Consultancy Service a note of the facilities they have used together with a brief description of what they are doing. However, users should always remember that these facilities are only available in some sense 'at their own risk'.

Macros currently available -

CHAP

LOCATE (exceptions:- volume serial numbers obtained by this macro cannot be relied on as being accurate)

RDJFCB

OBTAIN (the information obtained from this macro cannot be relied on as being accurate except in the short term. Please seek advice before using this macro).

DEVTYPE

Programmers should not refer to store other than the store explicitly obtained by them lest inconsistent results occur. Note also that store is not zeroed before use.

Experienced programmers who wish to use other macros are asked to consult the Computing Service.

BADFILE

BADFILE is a Phoenix flag which is set by the system and may be tested by the user. It is set if a dataset allocation error occurred in the last non-control command and is unset when the next non-control command is obeyed. See Section 4.10.2 for further details.

EXAMPLE

```
IF BADFILE QUIT 999
```

If a dataset allocation error occurred in the last non-control command, control returns to the calling level of command with a return code of 999 set.

BADP

BADP is a Phoenix flag which is set by the system and may be tested by the user. It is set if there is an error in a parameter occurrence, and is unset when the next non-control command is obeyed. See Section 4.10.2 for further details.

EXAMPLE

```
IF BADP ABANDON
```

If there is an error in a parameter occurrence the BADP flag will be set and so control is transferred to the console or the job step is terminated. Otherwise the command stream continues to be obeyed.

BADPARM

BADPARM is a Phoenix flag which is set by the system and may be tested by the user. It is set whenever there is an error in the parameter declaration for a C command, and is unset when the next command is obeyed. See Section 4.10.2 for further details.

EXAMPLE

```
IF BADPARM QUIT 99
```

If there is an error in a previous parameter declaration the BADPARM flag will be set and so control is transferred to the calling command. Otherwise the command stream continues to be obeyed.

BADSYNT

BADSYNT is a Phoenix flag which is set by the system and may be tested by the user. It is set if a syntax error occurred in the last non-control command and is unset when the next non-control command is obeyed. See Section 4.10.2 for further details.

EXAMPLE

```
IF BADSYNT QUIT 201
```

If a syntax error occurred in the previous non-control command the BADSYNT flag will be set and so control is returned to the calling level of command with a return code of 201 set.

BCPL

BCPL is a portable programming language that is suitable for compiler writing and other large systems programming applications. It is implemented on all the computers in the Computer Laboratory as well as at several other installations in this country and abroad.

REFERENCES

Example Sheet 16

'The BCPL Programming Manual' - by M. Richards

'The BCPL library for the IBM 370' - by J. K. M. Moody
(Available soon)

PROCEDURES

The following JCL and Phoenix procedures are available:-

BCPLC
BCPLCL
BCPLCLG
BCPLLG

Note that the Phoenix procedures cannot be used in a Phoenix console session. One should also remember that single segments of BCPL can be executed immediately after compilation using BCPLC and the 'Load-Go' Code-generator option This is usually the most convenient and economical way of running small BCPL programs since it eliminates the need for the Link-edit and Go steps.

PHOENIX EXAMPLES

(a) BCPLC PROGRAM=.ABC REGC=130K PARMC='S/KNGXXX'

compiles and runs the program .ABC producing a line numbered listing. The string XXX is passed as the parameter string to the running BCPL program Option S produces a source listing; K causes compilation of procedure call counts; N disables the generation of an object module and G invokes the load-go option.

(b) BCPLCL PROGRAM=.ABC NAMEL=.MODS(ABC)

compiles the program .ABC storing the resulting load module in .MODS(ABC).

(c) BCPLCLG PROGRAM=.ABC ENVG='DATA=.INPUT'

compiles, link edits and executes the BCPL program .ABC, supplying data from the dataset .INPUT using ddname DATA.

BCPL 2

PHOENIX PARAMETERS AND DDNAMES

REGC, REGL and REGG, and SECC, SECL and SECG control the region sizes and times for compilation, link edit and execution steps respectively in the usual way PRINTC, PRINTL and PRINTG control the destination of the SYSPRINT streams for the three steps.

The source program must be specified using the parameter or ddname PROGRAM The compiler options may be supplied using PARMC. The default setting is L12000/K which specifies a compiler list space size of 12000 and that function call counts are to be compiled.

The file inserted by the get directive GET "LIBHDR" can be specified using the parameter LIBHDR The default is the standard library header file BCPLIB LIBHDR

The size of the compiler's input/output and overlaying workspace can be specified using the parameter IOSPACE (default 54K).

The version of the compiler used can be specified using LIBC. The default value is BCPLIB.SYS2 but the backup version of the compiler, BCPLIB.SYS3, may sometimes be useful.

The SYSLIB library in the link editing step is the concatenation of the file specified by the LIBRARY parameter and BCPLIB.LIB.

JCL EXAMPLES

(a)
```
// EXEC BCPLCLG
//BCPL SYSIN DD DSN=MR10.TEST,DISP=SHR
//GO SYSIN DD *
73  P  42  X  Q
//*
```

compiles and executes the BCPL program held in MR10.TEST with the instream data made available on the standard input stream SYSIN.

(b)
```
// EXEC BCPLCL,PARMC='H/K',MAPL=MAP,NAME='SPQR.LIB(AB)',DISPL=OLD
//BCPL.SYSIN  DD  *
NEEDS "PM"
GET "LIBHDR"
. .
//*
//LKED LIB  DD  DSN=XYZ1 MODS,DISP=SHR
//LKED SYSIN  DD  *
    INCLUDE LIB(A,B,C)
//*
```

compiles a program and uses the link editor control statements defined by dd statement LKED SYSIN to include the modules stored in the library XYZ1 MODS. The output load module is stored in the existing library SPQR LIB

BCPL
3

(c)
```
//  EXEC BCPLC,LIBC=BCPLIB.SYS3,REGC=120K,
//      PARMC='S/KNG'
//BCPL.SYSIN  DD  DSN=MR10.TEST,DISP=SHR
//BCPL.DATA   DD  DSN=MR10.DATA1,DISP=SHR
//           DD  DSN=MR10.DATA2,DISP=SHR
//BCPL.SYS    DD  DSN=MR10.OVERLAYS,DISP=SHR
//BCPL.OUT    DD  DSN=MR10.OUT,DISP=(,CATLG),UNIT=DISC,
//      SPACE=(TRK,(10,5),RLSE),DCB=VBL
```

compiles and runs the BCPL program MR10.TEST using the load-go option in a region of 120K. The backup compiler BCPLIB.SYS3 is used for the compilation. The running program uses three datasets with ddnames DATA, SYS and OUT.

JCL PARAMETERS AND DDNAMES

The source program must be on an additional dd statement of the form:

```
//BCPL.SYSIN  DD  ...
```

The compiler options string is supplied using the parameter PARMC (it is a common mistake to use PARM). The go step parameter string can be specified using PARMG. A full list of available parameters for each procedure may be obtained by printing the dataset INFO.JCL.PARMS(procedure-name).

C

C is a Phoenix command which calls a Phoenix command sequence contained in a dataset, with optional substitution of parameters in the command sequence. COMMAND is a synonym for C.

EXAMPLES

(a) C

The commands in the current document are executed.

(b) C .B X=10

The commands in .B are executed. The value 10 is substituted for any occurrence of the parameter X in the command sequence contained in .B.

(c) C .CL:RUN + Y ABCD -Z Q=12

The parameter declaration sets the local parameters Y and Q, and unsets Z locally. The + is used to pass any other parameters defined locally at the previous calling level into the command sequence.

PHOENIX COMMAND SYNTAX

(a) C dsd[1] (P / PARS)[1] parameter-declaration[1]

(b) C dsd[1] (command-environment) parameter-declaration[1]

NOTES

The P or PARS keyword may be optionally followed by a parameter declaration which is used to set local parameters in the command sequence.

The P or PARS keyword is only needed if there is no dsd or if the first parameter in the parameter declaration is P or PARS.

For further details of the C command see section 4.8.1.

---

1 Optional.

CC

For details of the CC flag see RC.

CHAR

SET CHAR is a Phoenix command which enables various characters to be defined as different characters from the default ones. The special characters are those used for bracketing parameter occurrences (defaults < and >), the character used for delimiting certain strings (default '), and the line concatenation character (no default). The scope of the defined characters is to the end of the command sequence.

PHOENIX EXAMPLES

```
SET CHAR END !

SET CHAR BRA (

SET CHAR KET )

SET CHAR QUOTE ?

SET CHAR BRA [, END !

SET CHAR -END
```

PHOENIX COMMAND SYNTAX

```
SET CHAR [keyword character][,keyword character[3]]
   or
SET CHAR -END
```

Keywords

END — when the character is encountered, the line it is on and the next line are concatenated with the concatenation character omitted. The maximum length of a line is 255 characters. The line continuation character may be unset by use of the SET CHAR -END command.

BRA — the character specified replaces the default > as the opening bracket of a parameter occurrence.

KET — the character specified replaces the default > as the closing bracket of a parameter occurrence.

QUOTE — the character specified acts as a string delimiter, instead of the default character '.

---

3 may be repeated.

CLEAR

CLEAR is a Phoenix control command which cancels all previously set parameter values, resetting the preset parameters U and %USER to the current userid (the logged in userid or the job statement userid), and S and %SET to the current SET name. All other values are unset.

NOTE

CLEAR and SET CLEAR have the same effect.

EXAMPLES

```
SET CLEAR

CLEAR
```

COBOL

COBOL is a commercially oriented high level programming language of very limited use in a scientific environment. The version supported is ANSI-2 and is available via the JCL catalogued and Phoenix procedures COBUC, COBUCL, COBUCLG and COBULG. Further details of the language and the ANSI compiler may be found in the IBM manuals 'USA Standard COBOL Programmers Guide' (GC28-6399) and 'Full American National Standard COBOL' (GC28-6396).

Note that the Phoenix procedures may not be used in a Phoenix console session

PHOENIX EXAMPLES

(a) COBUCLG (PROGRAM %H#) LISTC=SOURCE
.
#

The program is given on a %H dataset, and the symbolic parameter LISTC is used to produce a source listing.

PHOENIX PARAMETERS AND DDNAMES

The source program must be defined by either the symbolic parameter or the ddname PROGRAM A complete list of the other symbolic parameters used in the Phoenix procedures may be obtained by using the command 'HELP PROC procedure-name', but the most important is probably the LISTC parameter (default value NOCLIST) which may be used to control the various listing options (for example SOURCE, CLIST and VERB)

JCL EXAMPLES

(a)
```
// EXEC COBUCLG
//COB SYSIN DD DSN=SPQR.PROG,DISP=SHR
```

This is the compilation, link editing and run of a COBOL program contained in the disc dataset SPQR.PROG

(b)
```
// EXEC COBUCLG,LISTC=SOURCE
//COB SYSIN DD *
.
//*
```

The program is given as an instream dataset and a source listing is produced by the use of the symbolic parameter LISTC.

JCL PARAMETERS AND DDNAMES

The complete list of symbolic parameters used in each catalogued procedure may be listed from the partitioned dataset INFO.JCL.PARMS using the member name corresponding to the procedure name.

COLLECT

COLLECT is a Phoenix command which may be used to collect the printer output from a 'saved' job (see SAVE). To collect saved output, the user must have correct jobqueue access status (see PASSWORD). By default output is typed and also copied to the current dataset. When the printer output has been collected, any other unit record output (punch or plotter) is queued for output to the appropriate destination unless READ has been used as an option for the command.

PHOENIX EXAMPLES

(a) COLLECT 123

Job number 123 is collected and the printer output sent to the results dataset (%R) which is copied to the current dataset (%C) and typed. Any other unit record output is queued for output as appropriate.

(b) COLLECT 456 USER SPQR PW SECRET TO .RES

The printer output from job 456 for user SPQR is filed to the dataset .RES.

(c) COLLECT 987 READ

The output from job 987 is typed and copied to the current dataset but the job is not deleted from the queue.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
JOB	integer		-	Mandatory item
USER	text	value of U or USER	-	-
(PW / PASSWORD)	text	value of PASSWORD or PW	-	-
TO[5]	dsd	%R	/N	-
READ	-	-	-	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

COLLECT 2

NOTE

(a) The typed output from the command can be interrupted without loss of information in the current dataset by pressing 'break' after the message

Jn SPQR collected

COMMENT

COMMENT is a Phoenix command which may be used to display text. The command is followed by text which is output to the command reflection dataset when the command is obeyed. COMMENT is unaffected by any of the switches.

PHOENIX EXAMPLES

COMMENT NOW AT LEVEL ONE

When this is obeyed the words 'now at level one' are output to the command reflection dataset which would be the console during a console session and the printer during a job.

PHOENIX COMMAND SYNTAX

COMMENT text

COMPARE

COMPARE is a Phoenix command and JCL catalogued procedure which is designed to check two sequential character files for altered, inserted or deleted lines. Two modes are available; difference mode produces a printout of the differences between the two files, and edit mode produces the edits necessary to change the first file into the second.

PHOENIX EXAMPLES

(a) COMPARE .A WITH .B

Two files .A and .B are compared. The differences between them are printed.

(b) COMPARE .A WITH .B OPT E TO .C

The files .A and .B are compared, and the edits needed to change .A into .B are filed to .C

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
FILE	dsd	%C	/R	Mandatory item if WITH absent
WITH	dsd	%C	/R	Mandatory item if FILE absent
OPT	text	-	-	-
STORE	(integer / integer K)	120K	-	-
TO[5]	dsd	*	-	-

JCL EXAMPLES

(a) // EXEC COMPARE,FILE='SPQR.A',WITH='SPQR.B'

Two files SPQR.A and SPQR.B are compared and the differences between them printed.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Key may not be omitted.

COMPARE 2

(b)
```
// EXEC COMPARE,TO='SPQR.RES',FILE='SPQR.A',OPT=E
//FILE2 DD *
.............
//*
```

The file SPQR.A is compared with an instream dataset, and the edits needed to change the file SPQR.A into the instream dataset are filed to the dataset SPQR.RES.

JCL PARAMETERS AND DDNAMES

The files to be compared are defined by the symbolic parameters FILE and WITH, or by the ddnames FILE1 and FILE2. Output is sent to the dataset defined by the symbolic parameter TO or by the ddname SYSPRINT. Options are passed to the the program using the symbolic parameter OPT.

OPTIONS

Options may be separated by spaces or commas.
The following options are available:-

B	Verification type messages suppressed.
D	Set difference mode. Default option.
E	Set edit mode.
L+	Check for the equality of line lengths.
L-	Do not check for the equality of line lengths. Default option.
Mn	Search for up to n mismatching lines (see note c).
Pn	Set printer limit of n. This prevents an excessive number of lines being printed. Default value is 999999.
Rn	Require n (>0) lines to restore equality after a mismatch (see note c). Default value is 5.
V+	Verify in normal format. Not allowed in edit mode. Default option.
V-	Do not verify in normal format. Not allowed in edit mode.
Wn	Truncate all lines after n (>0) characters. n does not include the carriage control character. Default value is 132.
X+	Verify in hexadecimal. Not allowed in edit mode.
X-	Do not verify in hexadecimal. Not allowed in edit mode. Default option.
Z$	Change the terminating character to some character $. Only allowed in edit mode.

COMPARE
3

NOTES

(a) Spaces are permitted anywhere except within an integer. Commas are permitted between options.

(b) Record lengths are verified only with L+ V+ X-

(c) M and R are used to control recovery after mismatches. Up to M lines of either or both files may differ; if no match is then found an error occurs. It will allow n1 lines to differ from n2 if there are at least

MIN(R,MAX(n1,n2)+1)

matching lines immediately afterwards. The default value for M is calculated to be the largest value that will fit into the available store

(d) Datasets produced by COMPARE may not contain carriage control characters if they are to be used as input to the Editor.

(e) The window size W does not include the carriage control character.

RETURN_CODES

The following return codes are issued:-

0 - the files are exactly the same.
4 - the files are not exactly the same.
8 - no match could be found.
12 - the parameters are invalid or insufficient store.
16 - a ddname does not exist.

COMPRESS

COMPRESS is a Phoenix command and JCL catalogued procedure which is used to eliminate the dead space left in a catalogued pds after members have been replaced or deleted. Members are moved to a new disc file which is then renamed with the original dataset name. The IBM utility IEBCOPY (see IBMCOPY) may be used for the same purpose.

PHOENIX EXAMPLES

(a) COMPRESS .PDS

.PDS is compressed to a new dataset with default size of SMALL.

(b) COMPRESS XYZ1.LIB VER=&D OPT=LA

XYZ1.LIB is compressed. The new pds is created with the LARGE size characteristic. Verification messages are filed to a temporary dataset.

(c) COMPRESS NAME .LINKLIB OPT TRK6,2,20

.LINKLIB is compressed. The new pds has a primary allocation of 6 tracks, a secondary allocation of 2 tracks and 20 directory blocks.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Note
NAME	dsd	-	-	mandatory item
OPT	text	S	-	see options below
VER[5]	dsd	*	/N	-
STORE	(integer / integer K)	20K	-	-

Choose any one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

COMPRESS
2

PHOENIX OPTIONS

The OPT parameter or keyword may be used to set the size of the destination dataset, or to specify CONTIG or RLSE. See Section 4.3.3 for details of the size qualifiers available.

JCL EXAMPLE

```
// EXEC COMPRESS,NAME='ABC1.PDS',SPACE='4,2,5'
```

Simple compress.

JCL PARAMETERS AND DDNAMES

The symbolic parameter NAME defines the dataset to be compressed, and SPACE the size of the new dataset as primary, secondary and directory allocation.

NOTE

When COMPRESS runs successfully the number of directory blocks used is printed, together with a message

'Compress succeeded'

RETURN CODES

The following return codes are issued:

0 no errors

8 Invalid option quoted
TTR for a member was invalid

12 Insufficient store was available
DCB characteristics were invalid

16 FROM or TO could not be opened

CONDENSE

CONDENSE is a Phoenix command and JCL catalogued procedure which copies or merges a pds to a new or existing pds removing any dead space. The IBM utility IEBCOPY (see IBMCOPY) may also be used for these purposes.

PHOENIX EXAMPLES

(a) CONDENSE .LIB .LIB2

condenses the pds .LIB to .LIB2.

(b) CONDENSE FROM .PDS TO .SAVE.PDS/N/LA

condenses .PDS to a pds .SAVE.PDS which is created as large if it does not already exist.

(c) CONDENSE .MODLIB TO .NEWLIB VER=&D OPT=R

merges the pds .MODLIB into .NEWLIB with replacement of members. (Option R - see below). Messages are sent to a temporary dataset.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Note
FROM	dsd	-	/R	mandatory item see note
TO	dsd	-	/N/PDS/S	mandatory item
VER	dsd	*	/N	-
OPT	text	-	-	see OPTIONS below
(WITH SYSIN)	dsd	%N	/R	see OPTIONS below
STORE	(integer / integer K)	20K	-	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

CONDENSE
2

JCL EXAMPLES

(a)
```
// EXEC CONDENSE
//FROM DD DSN=ZYX1.LIB,DISP=SHR
//TO DD DSN=ABC9.SAVE.LIB,DISP=(,CATLG),
// SPACE=(TRK,(5,1,2)),UNIT=DISC
```

condenses from ZYX1.LIB to ABC9.SAVE.LIB.

(b)
```
// EXEC CONDENSE
//FROM DD DSN=MNO1.SAVE.LINKLIB,DISP=SHR
//TO DD DSN=MNO1.LINKLIB,DISP=OLD
//SYSIN DD *
EXCLUDE MEM1,MEM2
//*
```

condenses from MNO1.SAVE.LINKLIB TO MNO1.LINKLIB omitting some members.

JCL PARAMETERS AND DDNAMES

The dataset to be condensed is defined by a dd statement with ddname TO. The output dataset is defined by ddname FROM. Options may be passed to the program by the symbolic parameter OPT or in a dataset defined by a ddname SYSIN.

OPTIONS

The default option is A; the options available are:

A ADD	All members of TO are kept; only those members in FROM but not in TO are copied.
R REPLACE	All members of FROM are added to TO; members with identical names in FROM and TO are replaced.
C COPY	All members of FROM are copied; previous members of TO are lost.
FORCEDCB	The output DCB characteristics are not checked to see if they are compatible with the input characteristics.
E list EXCLUDE list OMIT list	The members named are not copied from the input dataset.
I list INCLUDE list SELECT list	Only the members named are copied.

CONDENSE
3

NOTES

(a) Options may be separated by space, comma, / or end of line.

(b) Membernames in the lists are terminated at 8 characters, or by a space, comma, / or end of line. Membernames which contain special characters may be included by preceding the name with a quote. The next 8 characters are then read without checks.

(c) ADD, REPLACE, COPY and/or FORCEDCB, if used, must appear before any option which has a member list.

(d) INCLUDE and EXCLUDE are mutually exclusive.

RETURN_CODES

The following return codes are issued:

0 - No errors.

8 - An invalid option quoted.
The TTR for a member invalid.

12 - Insufficient store available.
The DCB characteristics were invalid.

16 - Unable to open FROM or TO

COPY

COPY is a Phoenix command and JCL catalogued procedure for copying one or more sequential datasets which may be members of a pds but which must not be load modules. Datasets are copied block by block using BSAM.

COPY is faster than FILE for blocked datasets and should be used in preference to FILE whenever the blocking and record format are not being altered.

Both the source and the destination datasets MUST reside on disc or magnetic tape.

The DCB parameters RECFM, LRECL and BLKSIZE of the destination dataset are unconditionally set to those of the source dataset, but other DCB parameters (e.g. DEN) may be altered. The program deals with the following record formats: F, FB, FBS, V, VB, VBS, U, FA etc.. Magnetic tapes may also be copied using the utilities TLSCOPY (TLS tapes only) or TAPECOPY.

PHOENIX EXAMPLES

(a) COPY FROM .AFILE TO .BFILE

This is a simple copy of one disc dataset to another.

(b) COPY FROM .A,.B,.C TO .D,.E,.F

Three disc datasets are copied.

(c)
```
JOB PQRS 1111,TAPE9 PS5601
PHOENIX
//TO1 DD DSN=PQRS.FILE1,UNIT=TAPE9,
//  VOL=SER=PS5601,DISP=(,KEEP),LABEL=1
COPY FROM .AFILE
```

This copies a disc dataset to magnetic tape.

(d)
```
JOB PQRS 2222,TAPE9 PS5601 PS5602
PHOENIX
//TO2 DD UNIT=TAPE9,VOL=(PRIVATE,RETAIN,SER=PS5601),
// DISP=(,KEEP),LABEL=2,DSN=PQRS.FILE2
//TO1 DD UNIT=TAPE9,VOL=(PRIVATE,RETAIN,SER=PS5601),
// DISP=(,KEEP),LABEL=3,DSN=PQRS.FILE3
COPY FROM .BFILE,.CFILE OPT STOP
```

Two disc datasets are copied to tape. If an error occurs the command will be abandoned.

(e)
```
JOB SPQR 333,TAPE9 ABC101
PHOENIX
//TO1 DD UNIT=TAPE9,VOL=SER=ABC101,LABEL=1,
// DISP=(,KEEP),DSN=PQRS.FILE1
SETUP (FROM1=.A)
COPY
```

This copies a disc dataset to tape.

COPY
2

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Note
FROM[5]	dsdlist	%N	/R	see below
TO[5]	dsdlist	%N	/N	see below
OPT[5]	text	-	-	-
VER[5]	dsd	*	/N	-
STORE[5]	(integer / integer K)	16K	-	-

Note:

Up to nine dataset names separated by commas may follow the FROM and TO keywords; these are set up with ddnames FROM1, FROM2 etc. and TO1, TO2 etc.

JCL EXAMPLES

(a)
```
JOB PQRS 123,TAPE9 XYZ234
JCL
// EXEC COPY
//TO DD DSN=PQRS.FILE1,UNIT=TAPE9,
// VOL=SER=XYZ234,DISP=(,KEEP),
//  LABEL=1
//FROM DD DSN=XYZ2.FILE,DISP=SHR
```

This is a simple copy of a disc dataset to tape.

Choose one of the alternatives in brackets.
Any item or key may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

COPY
3

(b) JOB SPQR 999,TAPE9 ABC101
JCL
// EXEC COPY,OPT='STOP'
//TO DD DSN=ABC1.DISCFILE,DISP=(,CATLG),
// SPACE=(TRK,(3,2),RLSE),UNIT=DISC
//TO2 DD DSN=ABC1.LIBRARY(PROG),DISP=OLD
//FROM DD DSN=PQRS.FILE2,UNIT=TAPE9,DISP=OLD,
// LABEL=2,VOL=(,RETAIN,SER=ABC101),
//FROM2 DD DSN=PQRS.FILE3,UNIT=TAPE9,DISP=OLD,
// LABEL=3,VOL=SER=ABC101

Two tape files are copied to disc. If an error occurs during processing the JCL step will be terminated.

JCL PARAMETERS AND DDNAMES

A dataset to be copied is identified by a dd statement whose name consists of FROM followed by a string of up to 4 alphanumeric characters (which may be the null string).

A destination dataset is identified by a dd statement whose name consists of TO followed by the same alphanumeric string that appears on the corresponding FROM dd statement.

Error messages are sent to the dataset defined by the symbolic parameter VER or ddname SYSPRINT. The symbolic parameter OPT may be used to halt processing after an error occurs.

NOTES

(a) The DCB parameters LRECL, RECFM and BLKSIZE in a TO dd statement, if specified, are ignored. Datasets are copied according to the EBCDIC order of the FROM dd statements, regardless of their order or method of specification. Thus, FROM precedes FROMB which precedes FROM1.

(b) The symbolic parameter OPT may be set to STOP to halt processing after an error occurs.

RETURN CODES

The following return codes are produced:-

If SYSPRINT cannot be opened, COPY stops immediately with return code 16.

Otherwise the final return code given is the highest over all copy operations, of the following:

0	Successful copy.
4	A TO data definition was missing or a FROM or TO data definition referred to a device other than disc or tape.
8	Insufficient store for the I/O buffer was given.
12	I/O error on FROM or TO dataset.

STORE REQUIREMENTS

The program uses two buffers, each of length equal to BLKSIZE. The default region size of the catalogued procedure (16K) is sufficient for copying datasets with BLKSIZE not exceeding 4K.

CREATION

CREATION is a Phoenix command which allows the manager of a filespace to change the default protection status for new permanent files. The creation status for new filespaces is FF. The protection status of an existing dataset may be changed using the PROTECT or EXAMINE commands.

PHOENIX EXAMPLES

(a) CREATION FR

The creation status is 'free' for the fileowner manager and his relations and is 'read only' for other users.

(b) CREATION UR FOR SPQR

The creation status for fileowner SPQR is 'update' for himself and his relations and 'read only' for other users.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
STATUS	text	-	-	Mandatory item
FOR	text	set name	-	-
VER[5]	dsd	*	/N	-

NOTES

(a) There are three protection levels:

F - Free access
U - Update; the dataset may be updated but not deleted
This is particularly useful for partitioned datasets
R - Read only

(b) There are two groups of users recognised by the protection system:

1. The filespace manager (normally the fileowner) and the filespace relations, as created by the RELATION command.

2. All other users.

(c) Only the filespace manager may change the creation status.

Any item or key may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

CROSSREF

CROSSREF is a Phoenix command and JCL catalogued procedure which produces a cross-reference table of the identifiers in an Assembler program, either by line number, or by sequence field (columns 73-80). It will accept IBM 360/370, Interdata, PDP11, GEC2050 or Calcomp 900 assembly language. It does not cross-reference operation codes. The input can have any record format.

PHOENIX EXAMPLES

(a) CROSSREF .A

A disc dataset is cross referenced and output sent to the printer or the console.

(b) CROSSREF FROM &A TO .A OPT C

This cross references a Calcomp assembly code dataset with output sent to a disc dataset.

(c) CROSSREF %H! OPT=S
program
!

This cross references an instream program by sequence field.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
FROM	dsd	%C	/R
TO	dsd	*A	/N
OPT[5]	text	I,L,R=3	See OPTIONS below
STORE	(integer / integer K)	60K	-
TITLE	text	name of input file	

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

JCL EXAMPLES

(a) `// EXEC CROSSREF,NAME='ABC1.PROGRAM'`

A catalogued dataset is cross referenced.

(b)
```
// EXEC CROSSREF,NAME='CARD DECK',OPT=S
//SYSIN DD *
 card deck
//*
```

This cross references a card deck by sequence fields.

JCL PARAMETERS AND DDNAMES

If the input is a catalogued dataset, it is defined by the symbolic parameter NAME in the calling statement. If not, the dd statement SYSIN in the catalogued procedure should be overridden to define the input. When the symbolic parameter NAME is not used to define the input it can be used to supply a title. Options are supplied using the symbolic parameter OPT.

OPTIONS

The default options are I,L,R=3.

The options available are:

I,IBM	input is IBM/370 assembly code.
ID,INTERDATA	input is in Interdata assembly code.
P,PDP11	input is in PDP11 assembly code.
G,GEC2050	input is in GEC 2050 assembly code.
C,CALCOMP	input is Calcomp-900 assembly code.
L,LINENUMBER	cross reference by line numbers.
S,SEQUENCE	cross reference by sequence fields (columns 72-80).
R=n	alter storage ratio to n ($1 \leq n \leq 10$).
M,MULTISTREAM	causes the line count to be reset at the start of each concatenated dataset, and the number of the dataset to be appended to the line number in the print out.

STORE REQUIREMENTS

The program obtains as much store as possible from the user's region, then uses one n-th of it for storing names, and the rest for storing references. The default value of n is suitable for cross referencing by line-number, but if sequence numbers are used (where each reference needs more core) it may be necessary to increase n to obtain a better usage of core. The area for storing references cannot exceed 120K.

CROSSREF
3

NOTES

The default destination of the output is to printer, but this can be overridden by setting the symbolic parameters TO (Phoenix) and PRINT1 (JCL). The output consists of a table of identifiers and the line numbers (or sequence numbers) of the statements in which they are referenced References to statements in which an identifier is defined are followed by the character '+'. At the end of the table lists of identifiers which are used but not set, set but not used, or set twice, are printed

RETURN CODES

A return code of 16 is issued if insufficient store is available.

CURRENT

CURRENT is a Phoenix command which creates a current document from any input dsd.

EXAMPLES

(a) CURRENT .A

A new current document is created which is a copy of .A.

(b)
```
CURRENT  %H!
text
!
```

This command must be used instead of INPUT in a job.

(c) CURRENT %P OPT=M(1,72)

The previous current document becomes the new current document. Only the first 72 columns of each record are copied to the current document.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
FROM	dsd	%C	/R	mandatory value
OPT	text	-	-	see OPTIONS below
STORE	(integer / integer K)	32K	-	-

OPTIONS

For a list of the available options see FILE.

Choose one of the alternatives in brackets.
Any item or key may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

DELETE

DELETE is a Phoenix command and JCL catalogued procedure which deletes disc datasets. The Phoenix command may also be used to delete Phoenix ddnames and Phoenix temporary datasets. DELETE cannot delete a member of a pds (see XPDS) or a magnetic tape dataset but it may be used to delete a complete pds.

PHOENIX EXAMPLES

(a) DELETE FILE

.FILE is deleted

(b) DELETE .A, B,.C:V

Three files are deleted and the last deletion is verified.

(c) DELETE PROG:QUIET

PROG is deleted and any soft error messages are suppressed.

(d) DELETE A

The ddname A is deleted, leaving any associated datasets untouched.

PHOENIX COMMAND SYNTAX

DELETE NAME[1] [dataset-name:options][1] [,dataset-name:options][3]

JCL EXAMPLES

(a) // EXEC DELETE,NAME='RJS4 MY.FILE'

This is a simple delete of one file.

(b) // EXEC DELETE,NAME='RJS4.A,RJS4.B,RJS4 C.D:V'

deletes three files with verification in the final case.

---

3 may be repeated
1 optional

DELETE
2

JCL PARAMETER

The symbolic parameter NAME should be followed by a deletion list. NAMES is a synonym for NAME.

OPTIONS

The following options are available:

V	verification messages are typed
QUIET	soft error messages are suppressed i.e. 'filename DOES NOT EXIST' is not typed
ASIS	text read after this option is not converted to upper case

The default options are:

no verification, not ASIS
at a console not QUIET and in a job QUIET.

Explicit error messages are given in all cases, and the program looks for the next command and carries on with the next dataset given, except when an illegal character is found.

RETURN CODES

A return code of 12 is produced for a hard error.

NOTE

If an option has been set for a particular item, it remains set for subsequent items.

DISPLAY

DISPLAY is a Phoenix command which provides system information such as the set parameters and their values, the set file formats, the set space parameters etc.

EXAMPLES

(a) DISPLAY

Any set parameters, space parameters, flags, switches, forms and variables are displayed.

(b) DISPLAY FLAGS,PARS

Any set flags, switches and parameters are typed.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
OPT	keyword	ALL	-	see below
STORE	(integer / integer K)	-	-	-
(TO / VER)	dsd	*	/N	-

Keywords	Output from DISPLAY
(CHAR / CHARS)	settings of the BRA, KET and QUOTE characters.
COMMANDS	names of loaded libraries and commands contained in them.
FLAGS	all set flags and switches.
FORM	all set forms.
GLOBALS	all globally set parameters.
LOCALS	all locally set parameters.
LIBRARY	all loaded libraries.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

DISPLAY
2

PARS	equivalent to GLOBALS+LOCALS.
SPACE	all set space parameters.
SW	as FLAGS.
SWITCHES	as FLAGS.
VARS	the values of all set variables.
ALL	all the above except COMMANDS.

NOTE

Several options may be specified at the same time. Options should be separated by commas.

EDIT

EDIT is a sequential editor used via a Phoenix command or a JCL catalogued procedure, which moves forwards through a document or documents line by line. If the output dataset is a tape or disc dataset, and a work dataset is provided it is possible to 'rewind' the output and thus make more than one pass through the source

Input:- Input may be in any record format Concatenation of unlike datasets is supported In the absence of any 'S commands input is taken from a dataset with ddname FROM This is known as the primary input

Output:- Output may be in any record format If any of the DCB parameters RECFM, BLKSIZE or LRECL are not specified for a disc or magnetic tape dataset they are copied from the current input (JCL) or taken as the default format (Phoenix) Punch datasets (SYSOUT=B 5, 7 or 8) are always opened with RECFM=V, LRECL=254 This allows records longer than 80 bytes to be sent to the paper tape punch Records longer than 80 bytes which are sent to the card punch are truncated by HASP. Printer datasets (SYSOUT=A or C) are always opened with RECFM=VA, BLKSIZE=141, LRECL=137 unless the current input has machine carriage control characters and the output concerned is not a listing dataset, in which case RECFM=VM is used When editing on-line output is sent directly to the terminal by means of the TPUT mechanism and the maximum record length is 254 characters In the absence of any 'O commands, output is sent to a dataset with ddname TO This is known as the primary output

Carriage control characters - EDIT will add or remove carriage control characters from the output dataset if required but it cannot change from one sort of carriage control character to the other If the type of carriage control characters is different for the input and output dataset EDIT behaves as if the input had no carriage control characters, and the default character (advance one line) is used for the output Special commands are provided for verifying and altering carriage control characters, they are not treated as part of the data in a line

Commands:- Editing commands are taken from the PARM field (if it exists), treated as a one-record dataset, followed by the dataset with ddname EDITS The 'C command can be used to read editing commands from other datasets If there are no commands the action of EDIT is to copy FROM to TO Commands are obeyed in textual order

Messages:- Verifications of the current line and error messages are sent to a dataset with ddname VER This is not opened until it is required

Additional outputs:- Copies of the output document, with or without line numbers, can be obtained by means of the P PL and PS commands

EDIT
2

Trailing spaces:- Trailing spaces are removed from all records by default both on input and on output except that fixed length output records are padded with spaces if necessary. The 'TR' command can be used to suppress this truncation The operating system produces inconsistent results if all records do not contain at least one data character. and so empty lines that are output in variable length or undefined record format are padded with one space character.

Multiple-pass operation:- The source document can be 'rewound' by means of the '*' command. This causes the rest of the current input to be copied to the output. and then an attempt is made to 'rewind' the current output and select it as input This will succeed if it is a tape or disc dataset If the primary output is rewound, it becomes the primary input the old primary input being closed (even if not exhausted) When TO is being used as input, the dataset with ddname WORK is used for the primary output If at the end of editing the primary output is found to be WORK, it is copied to TO This does not occur if editing is abandoned by using the 'Q' command.

PHOENIX EXAMPLES

(a) EDIT
edit commands
W

The current document is edited to produce an output document.

(b) EDIT A TO B
edit commands
W

A is edited with the commands from the terminal and the output goes to B

(c) EDIT C WITH D TO E VER=&F

C is edited with the commands in D and the output is filed to E. Verification and error messages are filed in a temporary dataset &F.

For more examples see Example Sheet 8.

EDIT
3

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Note
FROM	dsd	%C	/R	primary input
TO	dsd	%O	/N	primary output
(EDITS WITH)	dsd	*$	/R	edit commands
VER	dsd	*	/N	verification
WORK	dsd	%A	/N	work dataset
DD	dsdlist	%N	/R	for extra dsds
OPT	text	-	-	edit commands
STORE	(integer integer K)	32K	-	
ENV	text	-	/R	for extra ddnames

JCL EXAMPLES

(a)
```
// EXEC EDIT
//FROM DD *
 card deck
//TO DD DSN=ABC1 INPUT,DISP=(,CATLG),DCB=VB,
//   UNIT=DISC,SPACE=(TRK,(3,2),RLSE)
//EDITS DD *
 edit commands
//*
```

A card deck is edited and the output filed to ABC1.INPUT.

(b)
```
// EXEC EDIT
//TO DD DSN=ABC1 SOURCE PROGS(A),DISP=OLD
//FROM DD DSN=ABC1.INPUT,DISP=SHR
//MOREDS DD DSN=ABC1.EDITS(A),DISP=SHR
//EDITS DD *
 edit commands
C MOREDS
//*
```

The input dataset is edited with in-stream editing commands and also with EDIT commands stored on disc. The output dataset is filed to member A of ABC1.SOURCE PROGS For more examples see Example Sheet 3.

Choose one of the alternatives in brackets.
Any item or key may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

JCL PARAMETERS AND DDNAMES

The input dataset is described by a dd statement called FROM EDIT commands are defined by a dd statement called EDITS

Output is sent to a dataset defined by ddname TO Verification and error messages are sent to the dataset described by the dd statement called VER

EDITING COMMANDS

Most commands consist of one or two characters, possibly followed by arguments. although there are a few command names longer The possible editing commands are summarised on the 'Edit Reference Card' More than one command can appear on a line of input; commands are terminated by reaching the end of their arguments, if any. Commands which have no arguments are self-terminating If an error is detected while obeying a command the remaining commands on that line are not executed. The R command and the I command when followed by insertion material must be the last command on the line (although comment may follow). Lower case letters are synonymous with upper case in command names.

Spaces are not required to terminate command names, except when the argument is a ddname A space following the name of a command which has a character string argument will be used as the delimiter character for the string In general, spaces are only required as separators where the context would otherwise be ambiguous, and redundant spaces are ignored

Multiple execution:- Any command other than R or I with in-line insertion material can be repeatedly executed by preceding the command by a decimal count A sequence of commands can be repeatedly executed, provided the sequence is entirely on one line. by enclosing it in brackets and preceding the group by a decimal count Such bracketed groups may not be nested but may contain repeated single commands.

If a repeat count of zero is given for a command or group of commands, the group is repeated indefinitely until one of the following occurs.-

(a) The end of the current input is reached This does not cause an error, however the rest of the current line of editing commands is abandoned

(b) An error occurs Normal error handling which always includes abandoning the current line of edits, takes place

(c) BREAK is pressed when running in a Phoenix console session Normal BREAK handling takes place, that is, editing returns to interactive mode

(d) The command or command group has been executed 4294967296 times.

EDIT
5

NOTATION FOR COMMAND DESCRIPTIONS

The character '/' is used to denote a delimiting character for character strings used in context editing. The delimiter may be any character not included in the string, including space The final delimiter can be omitted if the command is the last item on a line of edits. The characters 's' and 't' are used to denote character strings. The character 'Φ' is used to denote any printing character. The characters 'n' and 'm' are used to represent unsigned decimal numbers. In addition, in all places where a line number is expected, '.' or '*' may be used to mean 'the current line number' or the end of the current input' respectively

A summary of all the commands follows Further details about the more complicated commands are given subsequently.

SUMMARY OF COMMANDS

A/s/t/	After string s in the current line, insert string t
B/s/t/	Before string s in the current line, insert string t
C ddname	Take subsequent editing commands from 'ddname' Revert back to the dataset containing the 'C ddname' command when "ddname" is exhausted.
CC/s/	Set the Carriage Control character for the current line to s, which must either be a single character or two hex digits. A warning message (return code 4) is given if the current output has no carriage controls. Control characters are assumed to be machine code carriage controls unless they are one of blank, +, -, 0-9, A, B, C, V, W in which case they are taken to be ANSI carriage controls.
CF ddname	Close File. This command can be used to close explicitly any source or output dataset other than EDITS VER, WORK, FROM or TO It is ignored if no dataset with the given ddname is found

EDIT
6

CL	Concatenate the next Line with this one and make the result the current line. The position of the character pointer is preserved.
Dn m	Delete lines n to m inclusive (m can be omitted) Line m+1 (or n+1 if m is omitted) becomes the current line.
DF/s/	Delete from the current line up to, but not including, the line beginning with string s.
DG/s/	Delete the Global exchange for string s; if s is null, delete all global exchanges in existence.
DL/s/	Delete from the current Line up to, but not including, the line containing string s.
DT/s/	Delete To All characters from the beginning of the (windowed) current line up to, but not including, the given context, are deleted.
E/s/t/	Exchange the first occurrence of string s in the current line for string t
F/s/	Find the line beginning with string s and make it the current line. Intervening lines are copied to the output.
G/s/t/	Globally exchange (all occurrences of) string s for string t as the source is scanned. s may not be null.
Hn	Halt at line n. The Editor will not move past line n until the 'ceiling' is reset. H* sets the ceiling to the end of the input dataset.
HAn	Halt After n lines; as H, but a relative number.

EDIT
7

In ddname	Insert the dataset 'ddname' before line n.
In <amending material> Z	Insert the amending material before line n. The terminating Z must be on a line by itself and may be either in upper or lower case. The terminator may be changed using the Z command, see below.
Jn	Justify from the current line to line n. (See description below). J* justifies to the end of the document.
JF/s/	Justify Find Justify from the current line up to and including the line beginning with string s
JL/s/	Justify Locate Justify from the current line up to and including the line containing string s.
JNn	Justify the Next n lines
K	Temporarily suspend any field setting(s) for the rest of this line of commands.
L/s/	Locate the line containing string s and make it the current line Intervening lines are copied to the output.
Mn	Move to line n and make it the current line.
MAn m	Set MArgins for the source document at columns n and m Characters outside these positions are removed on input.
MACn m	Set MArgins for the Command dataset.
MAO n m	Set MArgins for the Output (and listing) datasets
MAVn m	Set MArgins for the Verification dataset.
MFn1 n2 n3 ...ni	Mark Fields at columns n1, n2, n3, .. ni. (see later description).

EDIT
8

N	Move to the Next line and make it the current line.
O ddname	Send subsequent Output to 'ddname'.
P ddname	Send a copy of subsequent output to 'ddname'. Multiple copies (on different datasets) are permitted.
PA/s/	Point After. The left-hand edge of the window (the 'character pointer') is set so that the leftmost character in the window is the first character after the given context.
PB/s/	Point Before string s The character pointer is set to point to the first character in the string.
PE ddname	Send a listing of subsequent editing commands to 'ddname' (with line numbers).
PH/s/	Print Heading. The string is used as a page heading for output on listing datasets (i.e. those opened by the PL or PS commands). The date and time are added. If the string is null, heading printing is cancelled
PL ddname	As P, but with line numbers. (List)
PS ddname	As P, but with Source line numbers (TSAS format). Inserted lines are marked with '****' instead of a line number Deleted lines are indicated on the next line by -. Changed lines are indicated by +.
PO ddname	Print Off Suspend a previous P, PL or PS command for 'ddname'.
Q	Quit. EDIT stops straight away, no copying is done Control is returned to the terminal when Q is issued from a non-interactive file called interactively

EDIT
9

Qn	Quit if the current return code is >n (modulo 255) otherwise continue.
Rn m <amending material> Z	Replace lines n to m inclusive with amending material which need not be the same number of lines, m can be omitted. The terminating Z may be upper or lower case and must be the first character on a line and be followed only by spaces. The terminator may be changed using the Z command (see below).
Rn m ddname	Replace lines n to m with the contents of the dataset described by ddname. m may be omitted.
RCn	Set the Return Code to n (modulo 255).
RSn m	Restrict Searches (i e F, L, DF, DL, JF and JL commands) to character positions n to m inclusive. Note that RS is included in RV.
RVn m	Restrict View of source text to character positions n to m inclusive. Context editing commands operate within this "window".
S ddname	Read subsequent source records from 'ddname'.
SA/s/	Split After. Split the current line into two lines Characters before and including the string s are output as the first line, the remainder becomes the current line
SB/s/	Split Before Split the current line into two lines Characters before string s are output as the first, the remainder becomes the new current line.
SQ+	Sequence card image output Output line numbers to be inserted in character positions 73-80 of all 80 byte fixed length output records
SQ-	Switches off sequencing

EDIT
10

SQS — Source line numbers are inserted, with '****' for inserted lines (cf PS).

Tn — Type. Move on n lines through the source, verifying them T* types to the end of the input

TLn — Type List. As T but with line numbers

TR+/TR- — Set/unset TRail mode (see below).

V+/V- — Set/unset Verify mode

VC+/VC- — Verify Carriage controls The carriage control character for each line is typed immediately following the line number. The data portion of the line is typed on a new line

VE+/VE- — Verify Edits VE+ causes each line of edits to be echoed on the verification dataset when it is read

VW+ — Verify Window. The positions of both the left and right hand edges of the window are indicated underneath each verified line, except that, if the left (right) hand edge coincides with the left (right) hand end of the line, it is not shown.

VW- — (Default). No Verification of the Window takes place

VWL — Verify Window Only the Left hand edge is verified

W — Windup. Copy the rest of the current input to the output, then stop. Exhausting all command streams has the same effect

X+/X- — Set/reset escape mode In this mode lines of commands are scanned for escape combinations introduced by the character '@'. X+ and X- do not take effect until the next line of edits are read. For a list of standard escapes see Appendix A. Note that @X is not available Note that when edits are typed at a Phoenix console escape processing will be carried out by the communications system regardless of the mode set

EDIT
11

Z0	change the terminating character for the I and R commands to some printing character 0. If it is a letter, either case may be used.
?	Verify the current line.
!	Verify the current line, together with a second line indicating the case of the letters. Upper case letters are indicated by '%' and lower case letters by '$' in the second line.
+n	Move on n lines through the source document.
-n	Move on n lines through the source document, deleting them.
¦ or \	Ignore the rest of this line of command (comment facility).
*	Rewind. Copy the rest of the input to the output, close them both, then reopen the output and select it as the new input.
or >	Advance the current character pointer one.
or <	Reset the character pointer to its initial value.
:	Set the character pointer to the true end of the line (ignoring the window).
#	Delete the first character in the current window.
$	Force the first character to be in lower case if it is a letter; then advance the character pointer.
%	As $, but forces upper case.
=n	Set the current line number in the source to be n.

EDIT
12

'	Repeat the last command involving a character string.
"	Equivalent to N'.
&	Repeat the last L, F, DL, DF, JF or JL command. This command is in addition to the ' command, which repeats the last command involving a character string.

NOTES

1. OPERATIONS ON THE CURRENT LINE

All the commands for changing the current line operate on the characters of the line which lie within the current window. When a new current line is read in, the left hand and right hand window pointers are initialized to the values given in the last RV (restrict view) command. The default values for the window are 1,32768. If the first string in an A B or E command is null, it will cause the second string to be inserted at the start of the (windowed) line. The > command causes the left hand window pointer to be advanced by one, while the < command resets it to its initial value. The : command sets the left hand pointer to the true end of the line, and the right hand pointer to a large number. The $ and % commands change the case of the first character in the window (if it is a letter) and then advance the pointer; the # command deletes the first character in the window. Thus a series of these commands can be used to alter a line character by character. For example, if the current line is

sooty THE big cat sat on the

%>>>>>$$$>####:B// mat/ this line of commands changes it to

Sooty the cat sat on the mat

When the 'window' lies wholly within the current line, the number of characters inside it is kept constant by the insertion of spaces or the suppression of characters at the right hand side. An error message is given if any character other than a space is suppressed. If the left hand edge of the window is beyond the true end of the line and material is inserted, spaces are added to the line to bring it up to the required length.

EDIT
13

2. CONTEXT SEARCHES

The commands which search for a line containing a given context (L, F, JL, JF, DF and DL) operate within the character positions specified in the last RV (restrict view) command, or in the last RS (restrict searches) command issued after the last RV command; (i.e. RV includes RS). The 'search window' operates separately from the main window; and applies only to the F, L, JF, JL, DF and DL commands.

The search always starts at the current line, if any.

3. SPLITTING AND CONCATENATING LINES

The SB or SA commands can be used to split one line up into more than one line. The characters preceding (and including, if using the SA command) the given context are output as a complete record, and the remainder of the line becomes the new current line with the same line number as before. The search for the context takes place within the current 'window'. The CL command causes the next line to be read and appended to the end of the current line to form a new current line. The line number is increased by one. For the SA and SB commands, the window is reset as though the new current line had just been read in.

4. FIXED FORMAT FIELDS

The MF command is used to designate certain character positions in source lines as the start of fixed format fields. Up to 10 fields are allowed. For example, when editing Assembler source one might have MF9 15 35 72. When changes are made to the current line, the number of characters in each field is kept constant, if possible, by the addition or removal of spaces at the righthand side of fields after each alteration to the line. If a field overflows, a message is given, but the characters are not lost. (The next field is shifted to the right). The K command temporarily suspends the field settings during processing of the remaining commands on the same command line. MF with no arguments cancels all field settings.

5. GLOBAL EXCHANGES

Global exchanges are performed 'on the fly' as the editor moves through the source. The effect of a G command is therefore merely to add to the list of globals currently in force, and to carry out the exchange on the current line, if there is one. Global exchanges are performed in the order in which they appear; e.g.

g/a/b/ g/b/c	causes all occurrences of the string 'a' to be replaced by the string 'c'.
g/b/c g/a/b	causes all occurrences of the string 'a' to be replaced by the string 'b'.

EDIT
14

If a previous global exchange for the same string is in existence, it is cancelled.

g/a/b/ g/b/c/ g/a/d/	causes all occurrences of the string 'a' to be replaced by the string 'c' in the current line, but all occurrences of 'a' to be replaced by 'd' in subsequent lines.

Global exchanges can also be cancelled by the DG command, and all globals are deleted when a rewind occurs. A null first string is forbidden for the G command; for the DG command it means 'cancel all globals'. Verification of global exchanges is inhibited when the editor is in interactive mode and the current line number is advanced by means of the R or I commands with inline insertion material. This is to stop the typing of verifications interfering with the input of amending material. Command sequences such as M20 I. should be avoided when doing global exchanges interactively. Global exchanges are carried out inside the current window. In the case of F, L, JF, JL, DF and DL commands, the test for the required string is done after any globals have been applied. Global exchanges involving space characters should be used with great care if field or window settings are in force, as infinite loops can result.

6. MARGINS

The MA, MAC, MAO and MAV commands may be used to limit the width of the input, command, output and verification datasets respectively. The effect of the MA command is to remove any character outside the margins before editing on the current line begins. The effect of the MAO command on output lines is to insert n-1 spaces before the first character and to truncate lines to m-n+1 characters subject to LRECL constraints. After a rewind operation the margins are set back to their initial values for the source and output datasets. The initial setting of the margin is MAO1 32768 for the output dataset. The MAO command is also useful in conjunction with the justify commands.

7. JUSTIFICATION COMMANDS

Justification facilities in Edit are very simple.

The J, JN, JF, and JL commands are four different ways of specifying which lines are to be justified. In the case of the JL and JF commands, the search for the context takes place within the current 'search window' as each successive line is read. The basic result of justifying a set of lines is as though all the lines had been concatenated, with one space inserted between each, and the result chopped up again into the largest sized pieces that would fit into the output records, the new breaks always being made immediately after a space character which will normally be subsequently removed for V and U format output. No attempt is made to right justify, nor are spaces inserted or removed in the body of the text. There are also some additional features:-

(a) The length of the output records can be controlled by means of the MAO (output margins) command, subject to LRECL.

EDIT
15

(b) The RV (restrict view) command can be used to justify a piece of indented text with, possibly, text in the margin.
When a line is read which contains characters other than spaces to the left of the window, it is treated as the start of a paragraph and not 'flowed' onto the previous line. If there are non-blank characters to the right of the window, the line is output as it stands and not 'flowed' onto subsequent lines. In addition, all output lines that are not at the start of paragraphs are suitably indented. Start of paragraph lines are printed with the left hand margin field unaltered.

(c) A blank line, a line beginning with a space character (inside the window), or a line containing an ASCII carriage control character other than blank also indicates the start of a new paragraph.

Justification takes place immediately; it is not deferred like global exchanges. All output lines are verified if in verify mode, the line number being that of the most recently read input record. At a Phoenix console, BREAK may be used to interrupt any of the justification commands. Global exchanges are carried out as each new line of source is read and are not verified separately. Note that, when verifying indented text, the window set up with the RV command also applies to the JL and JF commands unless an RS command follows RV.

8. CONTROL OF DATASETS

When a C, R, I, or S command is issued, the dataset concerned, if any, must either be closed, or open as a non-current source or output dataset (except that the current source is allowed for S). If it is open for output, the dataset is closed before opening for reading; otherwise it is read from its current position.

A side effect of the S command is that the current line is output before the input is switched. Separate line counts are maintained for each dataset.

When an O command is issued, the dataset concerned must either be closed, or open as an output dataset, in which case subsequent output records are added to it.

When a P, PL, or PS command is issued, the dataset concerned must either be closed, or open as a listing dataset (i.e. previously opened by P, PL or PS). The output to listing datasets takes place concurrently with output to the main output dataset. All listing datasets, except those opened by a PE command, are automatically closed (and P, PL and PS commands cancelled) when a rewind occurs. Listing datasets (i.e. those opened by P, PL and PS commands) may be closed by the CF command.

EDIT
16

9. RESERVED_DDNAMES

Certain ddnames are treated specially by EDIT. They need not necessarily exist, but if they do they will be used as follows:-

(a) VER — This dataset is used for error messages and verification of the current line. It is not opened until it is needed, but if opening fails, editing is abandoned. When editing is in interactive mode, VER is always required.

(b) EDITS — If EDITS exists, it is opened at the start of editing, and commands are read from it when the PARM field is exhausted.

(c) FROM — If FROM exists, it is opened at the start of editing, and selected as the current input and the current primary input.

(d) TO — If TO exists, it will be opened as the current output and current primary output when an output is required and no other is selected, unless there has been a rewind and TO is the primary input.

(e) WORK — This dataset is used instead of TO as the primary output when TO is open as the primary input. Notwithstanding anything said elsewhere in this document, WORK is always opened for output with the same DCB parameters as TO.

In commands which take ddnames as arguments, EDITS, VER, and WORK may not appear, while FROM and TO may only appear in the S and O commands respectively, in which case they select the current primary input or output.

10. MODES

(a) The Editor is said to be in interactive mode when the current command stream is attached to a console. In this mode, successful execution of a line of commands which does not require verification of the current line causes a blank line to be sent to the verification dataset, to signify that the editor is ready for

EDIT
17

another line of commands. If the editor is in interactive mode at the start of editing, the message EDIT READY is sent to the verification dataset. Use of the C command can cause the editor to go into or leave interactive mode. Note that it is possible to run the editor with EDITS attached to a terminal, and VER referring to some other disc dataset. This case is not treated specially.

(b) Verify mode is set or unset by means of the V+ and V- commands. In this mode, the current line is verified when it is altered, otherwise verification occurs only in error situations. (See under 'verification'). The editor is initially in verify mode only if it is initially in interactive mode.

(c) Trail mode is set or unset by means of the TR+ and TR- commands. In this mode trailing spaces are not truncated from records. The editor is initially not in trail mode.

(d) Carriage control verify mode is set or unset by means of the VC+ and VC- commands. In this mode, the carriage control character is output immediately after the line number when a line is verified. For datasets with machine carriage controls, the hexadecimal value of the character is given. The editor is initially not in VC mode.

11. VERIFICATION

Verification of the current line can always be forced by the ? command. Automatic verification occurs in verify mode when a line which has been changed but not yet verified is output, except when this is due to an interactive R or I command with following insertion material. Automatic verification may also take place when a complete line of editing commands has been executed, depending on whether the current line has changed since the last verification, whether the editor is in verify mode, whether there has been an error, and whether the editor is in interactive mode as follows.

('-' means irrelevant)

	changed	interactive	error	what to do
	(-	yes	yes	nothing
unverify	(-	no	yes	verify
mode	(-	yes	no	blank line
	(-	no	no	nothing

EDIT
18

	changed	interactive	error	what to do
verify	yes	-	-	verify
	no	no	no	nothing
mode	no	no	yes	verify
	no	yes	yes	nothing
	no	yes	no	blank line

12. ERRORS

There are four classes of error, corresponding to return codes 4, 8, 12 and 16. A return code of 16 is given if the VER dataset is absent when required. A return code of 4 is given for minor errors (e.g. field overflow). A return code of 8 indicates a more serious failure (e.g. no match in a context command), while a return code of 12 is usually caused by a severe error in connection with opening or unexpectedly exhausting a dataset. An error message is sent to the VER dataset if it exists, and the rest of the current line of editing commands is abandoned. When the editor is not in interactive mode, the current line of editing commands is reflected, with a pointer to the last character of the command which provoked the error. The editor then reads the next line of commands, except in the case of a severe error (return code greater than 8). In this case the editor checks to see if it is in interactive mode. If it is not, it abandons the current command stream and reverts to the previous one (if there have been C commands). It carries on doing this until either it finds a command stream to terminal, or it runs out of command streams, in which case editing is abandoned (with a suitable message). A Q command is forced if an error with return code greater than 8 occurs, or if 'break' is pressed when commands are being obeyed from a non-interactive file.

13. RESTRICTIONS

(a) The sum of the lengths of the character strings in any command may not exceed 160.

(b) There may not be more than ten field boundaries specified in the MF (mark fields) command.

(c) A repeated group of commands must be all on the line, and may not contain other repeated groups.

EDIT
19

(d) The working space required for changing the contents of a current line is not obtained until it is needed; the amount of space obtained depends on the logical record length (LRECL) of the current input at the time. The maximum length to which current lines can subsequently be extended is twice this LRECL. (They may also be truncated on output, of course). The CC, $ and % commands do not cause creation of workspace.

(e) The maximum logical record length that the editor will handle is 32767.

14. STORE REQUIREMENTS

EDIT program	just over 10K, permanently resident in store.
Global workspace	1/2 K approx.
Workspace for each open dataset	150 bytes + 2*BLKSIZE
Workspace for each global exchange in existence	8 bytes + lengths of strings
Workspace for changing current line	4*LRECL of input (see 'restrictions')

In addition there is a fair amount of system overhead.

With FROM and TO referring to typically blocked disc files, EDITS from cards, and VER to printer, EDIT will run comfortably in the 32K which the catalogued procedure has as a default. If more datasets are to be concurrently open, the region size must be increased accordingly.

15 MISCELLANEOUS NOTES

(a) All commands which take numbers as arguments are terminated by a non-digit (except for or * if applicable). Thus a legal string of commands is

```
D1 20 D45 D69      | comment
```

However, EDIT always tries to find the longest command it can, so therefore

```
D1 4(E/A/B/  E/C/D/)
```

would delete lines 1 to 4 and then give a syntax error.

EDIT
20

Commands which take a ddname as an argument are terminated by a character which is not a letter, national character or digit. Thus

```
I24 INS 4(I 26 REP) S FRED=9
```

is legal

(b) A very easy mistake to make is leaving out the final delimiter of a context command which is the last command in a repeating group, especially if the delimiter is a space, e.g.

```
5(N E A B)
```

(c) A repeated context command on the same line always starts searching for the character string at the start of the (windowed) line. Thus

```
4A/A/B/
```

will insert BBBB after the first A in the line, and so will

```
4E/A/AB/
```

This effect can be avoided by advancing the character pointer (i.e. the left hand edge of the window) after each exchange. The PB command can be used to point to the first occurrence of the string, e g

```
4(PB/A/  A/A/B/  2>) <
```

causes a B to be inserted after each of the first four A's in the line and resets the character pointer at the end.

(d) The restrictions on repeated groups of commands (namely, that they be all on one line, and not nested) can be overcome, at some cost, by careful use of the 'C' and 'In ddname' commands. For example, to insert text before each occurrence of a given context one could write

```
100(L/CONTEXT/I. INS N)
```

having previously set up the insert material on the dataset with ddname INS Alternatively, one could use

```
100C  DD2
```

where DD2 contains

```
L/CONTEXT/ I.
insert material
Z
N
```

EDIT
21

Repeat counts are associated with each dataset from which editing commands are being read, and so sequences like

```
5(L/A/ C  DD2  L/B/)
```

where DD2 itself contains repeated groups of commands, will work as expected.

Note that provided the dataset is set up in the JCL (offline) or in the command (under Phoenix), it can be used dynamically again and again for the purposes suggested above. The insert data or list of commands can be written to the dataset by a sequence like

```
O  DS                          | switch output to 'DS'
ZX                             | change terminator to X
I.
L/CONTEXT/I.
insert material
Z
X
O  TO                          | switch output back to 'TO'
```

The commands written to DS can now be obeyed by means of the C command.

EDSORT

EDSORT is a Phoenix command and JCL catalogued procedure which may be used for sorting and merging files of a subset of EDIT commands (see Notes). EDIT commands in a new dataset are sorted, then merged with any already sorted commands in an old dataset.

PHOENIX EXAMPLES

(a) EDSORT

sorts the EDIT commands in the current document. The sorted commands are written to a new current document.

(b) EDSORT .OLD.EDS NEW.EDS .ALL EDS

sorts .NEW EDS then merges this file with .OLD.EDS and puts the result in .ALL EDS.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
OLD	dsd	$	/R
NEW	dsd	%C	/R
TO	dsd	%O	/N
STORE	(integer / integer K)	36K	-

JCL EXAMPLES

(a)
```
// EXEC EDSORT,OLD='SPQR OLD EDITS',
//    NEW='SPQR.NEW EDITS',TO='SPQR.MERGE.EDITS'
```

sorts SPQR NEW.EDITS then merges with SPQR.OLD.EDITS. The sorted edits are filed as SPQR.MERGE.EDITS.

Choose one of the alternatives in brackets.
Keyed items may occur in any order; unkeyed items in the order given.
Any key or item may be omitted unless otherwise stated.

EDSORT
2

(b)
```
// EXEC EDSORT
//OLDED DD *
 old edits
//*
//NEWED DD *
 new unsorted edits
//*
//TO DD DSN=SPQR.EDITS(V4),DISP=OLD
```

merges two sets of edits and files the results.

JCL PARAMETERS AND DDNAMES

The previously sorted dataset may be defined by the symbolic parameter OLD or the ddname OLDED. The new unsorted dataset should be defined either by the symbolic parameter NEW or by the ddname NEWED. Output is to the dataset defined by the symbolic parameter or ddname TO. For a full list of symbolic parameters print the file INFO.JCL.PARMS(EDSORT).

NOTES

(a) The sorting and merging is done on a line by line basis, so editing commands which change the line number must always be at the start of a line.

Only the following commands (which must be at the start of a line) are recognised:

```
Mn
Dn (m)
In
In ddname
Rn
Rn (m) ddname
Yn (m)
```

For details of the first 6 commands see the EDIT Commands. The last command causes the (m+1)th EDIT command for line n to be deleted from the OLD dataset. m=0 by default.

(b) The rules for sorting EDIT commands for the same line are given by:-

		LAST EDIT				
		D	I	M	R	Y
FIRST EDIT	D	-	S	S	S	S
	I	-	-	-	-	S
	M	-	S	-	-	S
	R	-	S	S	-	S
	Y	-	-	-	-	-

EDSORT
3

S indicates that the EDIT commands are swapped, ie the last edit is put first.

(c) The rules for merging files are given by:

NEW EDIT	OLD EDIT D	I	M	R
D	D	O	O	D
I	N	O	N	N
M	N	O	O	N
R	D	O	O	D

D indicates that the old edit is deleted.

N indicates that the new edit comes first.

O indicates that the old edit comes first.

RETURN CODES

A return code of 16 is issued if

(a) there is not enough store available for sorting

(b) the input dataset with ddname NEW could not be opened

(c) the output dataset with ddname OLD could not be opened

ERRJUMP

SET ERRJUMP is a Phoenix command which specifies a label for error handling. Control will be transferred to this label if an error which sets the ERROR flag (see ERROR) occurs, and the next command is not IF or UNLESS. The label should be at the same command level, or confusion will result. See Section 4.10.7 for further details.

A local equivalent of SET ERRJUMP is also available. If the environment-element %ERRJUMP=label is present in an environment of a command and switch NOSTOP is unset and there are no succeeding conditional commands and the command terminates with an error state, then a jump to the given label will be taken. The effect of the %ERRJUMP is completely local to the command in whose environment it occurs and is therefore equivalent to a SET ERRJUMP label before the command, and a SET ERRJUMP * after the command. A %ERRJUMP takes precedence over any existing SET ERRJUMP.

PHOENIX EXAMPLES

(a)
```
SET ERRJUMP A
 ...
 ...
command sequence not
containing ERRJUMP
 ...
 ...
TYPE .FILE
QUIT
*A
** ERROR IN .FILE
SET ERRJUMP *
 ...
 ...
```

If an error occurs before or during the typing of .FILE a jump is made to A, and processing continues from that point. Otherwise the file is typed and control returns to the calling level of command.

(b)
```
...
:ABCD (EFGH=* IJKL=.K %ERRJUMP=MNOP)
QUIT
*MNOP
** deal with errors
...
...
```

PHOENIX COMMAND SYNTAX

Key	Valuetype	Notes
SET ERRJUMP	(label / *)	label to be jumped to. unsets the error jump.

Choose one of the alternatives in brackets.

ERROR

ERROR is a Phoenix flag which may be tested by the user. It is set whenever any of the flags ABEND, RC, BADPARM, BADP, BADSYNT or BADFILE are set. It is unset when the next non-control command is obeyed.

See Section 4.10.7 for further details.

PHOENIX EXAMPLE

```
*C
UNLESS ERROR SKIP A
SEND ABC1 ERROR AT C
QUIT
*A
```

If the ERROR flag is set control returns to the calling level of command, otherwise control is transferred to label A.

EXAMINE

EXAMINE is a Phoenix command and a JCL catalogued procedure that may be used to give details of the names and characteristics of catalogued and temporary datasets but not system datasets. It can give details of the space allocated to the datasets (i.e. the space requested by the SPACE parameters), the space used within this allocation, the total space both allocated and used, the time of the last modification of the dataset and the user by whom it was modified and the DCB characteristics. The income, credit and credit bound of a filespace may also be requested.

Note that accounting by the filing system is for the space allocated rather than the space used. (See Section 3.1.2).

EX is a synonym for the EXAMINE Phoenix command.

PHOENIX EXAMPLES

(a) EX RJS4/I

Interactive examine (see below).

(b) EXAMINE /U

This examines disc space usage, credit and income of the set userid in a console session or the userid quoted on the job statement of the Phoenix job.

(c) EXAMINE SPQR.A TO RJS4.A/N

The output, a list of files with first two dataset name components SPQR.A, is placed in a dataset RJS4.A.

(d) EXAMINE &TEMP

The temporary dataset &TEMP is examined.

(e) EXAMINE &

All the user's temporary files are examined.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
NAME	text	set name	-
TO[5]	dsd	*/FA	/N

Any key or item may be omitted unless otherwise stated.
5 Key may not be omitted.
Keyed items may occur in any order; unkeyed items in the order given.

EXAMINE
2

PHOENIX NOTE

If no userid has been explicitly set using SET, then the logged on userid (in a console session) or the userid under which the job is being run (in a job) is used.

JCL EXAMPLES

(a) // EXEC EXAMINE

This examines all the datasets belonging to the fileowner defined by the user identifier under which the job is being run. This is equivalent to specifying NAME='userid/A'.

(b) // EXEC EXAMINE,NAME='RJS4.A.B,RJS4.C'

The files mentioned are examined.

(c) // EXEC EXAMINE,NAME='&TEMP'

The dataset &TEMP is examined.

JCL PARAMETERS

If the optional NAME parameter is not set, the datasets of the userid within whose job the procedure is called are examined. If the NAME parameter is present it may be followed by a list of one or more values separated by commas.

OPTIONS

The text given to EXAMINE consists of a list of fileowners, partially qualified dataset names and fully qualified dataset names, each item being optionally followed by '/' and one or more options. Multiple options are each separated by '/', and multiple items by ','. The options available are:-

A or ALL	This option is equivalent to F/M/P/S/W. This is the default for single datasets and in all cases where EXAMINE is used with output not directly to a console.
B or BRIEF	The disc space and credit are typed.
C or CREATED	The date of creation is printed for each dataset.
DEL or DELETE	Delete file (see INTERACT).
E or EXPIRE	The expiry date of each dataset is printed (see note (c)).
F or FORMAT	The DCB information (such as organisation, record format, block size etc.) is printed for each dataset.
G	This option is equivalent to ALL/C/E/V. (See note (c)).

EXAMINE
3

H or HEADING	This causes output from EXAMINE to be in columns, with a heading for each column (default in the case of a fileowner or partially qualified dsname).
I or INTERACT	This is only allowed online when output is to the console. EXAMINE pauses after each line of information for input from the console. Keyword options may be separated by blanks, commas or '/' and are terminated by carriage return. The commands are: A, B, C, DEL (to delete the file), E, F, G, H, J, K, L (end interactive mode), M, P, Q (as defined below), S, T, U, V, W or Y. If carriage return only is typed EXAMINE moves onto the next dataset with all options unchanged.
J or JUSTIFY	As H, but each column is allowed enough space for the largest possible parameter rather than the largest that actually occurred. This is mainly useful for users who post-process EXAMINE output with their own command sequences!
K or KWIK	Each printed line has all multiple spaces removed. Brackets and commas are inserted in appropriate places to improve readability. All heading output is suppressed.
L or LIST	Ends interactive mode (see INTERACT).
M or MODIFIED	The date and time at which a file was last opened for output is printed. If this was before the system started recording these dates a row of asterisks will be printed instead.
N or NEWEST	The order of printing files for each fileowner will be by descending order of modification date and time, with the most recently modified first.
NO	See note (d).
O or OLDEST	As NEWEST, but with the oldest modified file first.
P or PROTECT	The protection status of each file is printed. In INTERACT mode, P may be followed by =xy, where xy is the required protection status. The status is set as requested, and the file is examined again to verify this.
Q or QUIT	Stop examining this fileowner, and proceed to the next item in the initial option list (see INTERACT).
R or RELATION	The identifiers of any users allowed other than public access to files in the fileowner's filespace are printed. Either files or relations may be examined with each command item, and if the R option is used, all others except B, DEL, H, I, K, L and Q are ignored.

EXAMINE
4

S or SPACE	The disc space allocated and used are printed for each dataset.
T or TITLES	The only information typed for each file is its name.
U or USAGE	No datasets are examined, but details of the fileowner's disc space usage, credit and income are printed.
V or VOLUME	The volume identifier of the disc containing each dataset is printed. (See note (c)).
WARN	EXAMINE checks the disc space allocated, income and credit figures, and adds a message starting 'WARNING' after the usage summary table if more space is allocated than the income allows, or if the usage is such that the filespace would be inhibited on the next working day unless some of the space is freed. This option is a default.
W or WHO	The identifier of the last user to modify or create each file is printed (if this happened before the system recorded this information a row of asterisks is printed).
X	The list of files is printed unsorted. For normal fileowners this is alphabetic order (the default). This is intended for use by users with access to private disc packs. (See note (c)).
Y	Certain useful information bytes from the DSCB (such as the byte containing the 'CONTIG' bit) are printed in binary for each file. (See note (c)).
Z	Each list of files is sorted in reverse alphanumeric order.
"string" or 'string'	Only filenames containing this string will be examined. If the string is preceded by 'NO', any filetitle containing the string is excluded.
CVOL=discid	The file catalogue on the specified volume will be used instead of the master system catalogue. (See note (c)).
TYPE	(In interactive mode only). The file will be typed at the console. The BREAK key may be used to interrupt the typing operation and re-examine the file (by typing 'X' after the ***-) or for a PDS to start typing the next member by typing just carriage return.
VOL=discid	The file catalogue will not be used, but a search for the specified file will be made on the specified volume. If no filetitle is specified, all files on the volume will be examined. The disc should be a private pack, and must already be mounted. 'VOL=' may be omitted if the first three characters of the disc identifier are SYS or PRI.
XPDS	The XPDS program is invoked to give further information about partitioned datasets. If the XPDS option was present in the command line, a list of member names is

EXAMINE
5

typed after each file that has partitioned organisation. If the XPDS option is entered interactively in response to a request for input from EXAMINE, XPDS will prompt the user itself with >, and the user may enter any XPDS command or commands (see the specification of XPDS). Typing just W in response to this request will produce just the member list, and EXAMINE will then prompt the user again (with the normal ?) for further EXAMINE options. In interactive mode, the XPDS option applies only to the file currently being examined.

RETURN_CODES

If error conditions are encountered, the return code is set to reflect the type of error. Possible return code values are:

0 - No errors.
4 - Fileowner, dataset name(s) or relations do not exist.
8 - Insufficient store available to complete the list.
12 - Syntax error in dsname, or one or more illegal options found.
16 - Error on TO dataset; or CVOL not mounted (see note (c)).
20 - I/O error while reading catalog.

NOTES

(a) The default option is ALL, except when EXAMINE is used directly at a Phoenix console when it is TITLES if a fileowner or partially qualified dataset name is given, or ALL/KWIK if one or more specific filenames are given. The WARN option is always selected in default.

(b) For each fileowner examined, a summary of space accounted and used, and details of his credit etc. are produced at the end of the file list if the SPACE or USAGE options are requested, or if no other data selection option (C, E, F, M, P, Q, R, T, V, W, Y) is specified (for example in the default case). This summary and any consequent warning messages can be suppressed in all cases by the NOUSAGE option.

(c) These options are likely to be of interest to system programmers only.

(d) If any of the keywords is preceded by 'NO' (which may not be abbreviated) the option is cancelled. Typing /* is equivalent to typing L. The A, C, DEL, E, F, G, M, P, S, V, W, Y, TYPE and XPDS options apply to the dataset which has just been examined. If any information is requested which has not been typed, the dataset is examined again.

FALSE

FALSE is a Phoenix flag which may be tested by the user. It is always set to false. It is useful when used in conjunction with a conditional statement.

PHOENIX EXAMPLE

```
UNLESS <A> STATUS
```

If the value of A is set to the string FALSE the STATUS command is obeyed, otherwise control is passed to the next command in the sequence.

FAST

FAST is a flag which may be tested by the user. It is set throughout a Phoenix Monitor job.

EXAMPLE

```
IF FAST SKIP ON
QUIT
*ON
SEND XYZ1 LIST JOB COMPLETED SUCCESSFULLY
```

This might be useful within a command sequence that could be executed either online or in a Phoenix Monitor job. It will only send the message if the sequence is obeyed in a Phoenix Monitor job.

FD

The SET FD command may be used to change the Phoenix default DCB qualifier from the usual VBL.

EXAMPLE

```
SET FD FB 80 3120
```

sets the default DCB to have record format FB, logical record length 80 bytes and blocksize 3120 bytes.

NOTES

Note that all three subparameters (record format, logical record length and blocksize) should be specified; the shortened forms of the DCB parameter cannot be used here.

FILE

FILE is a Phoenix command and JCL catalogued procedure for copying a sequential dataset (which may be a member of a pds). It can change the blocksize, record length and record format of the dataset and remove or add trailing spaces and/or sequence numbers as desired The program deals with all record formats. Carriage control characters may be present and are treated as characters in the record. COPY should be used when the DCB of a dataset is not to be changed

Load modules will not be copied correctly by FILE. CONDENSE. IBMCOPY or the Linkage Editor must be used instead

PHOENIX EXAMPLES

(a) FILE .A

files the current document to A.

(b) FILE FROM .B TO .C:D/S OPT=M(1,72)

files a sequential dataset as a member of a pds, removing sequence numbers.

(c) FILE TO &C/FN FROM .C:D

forces the output dataset DCB to be copied from that of the input dataset.

(d) FILE .A OPT=E

files the current document to .A interpreting escaped characters.

(e) FILE .A AS .B/FB

files a sequential dataset as a new fixed blocked catalogued dataset.

(f) FILE &C/FB FROM ABC

creates a temporary fixed blocked dataset from a dataset defined with the ddname ABC.

(g) FILE .A AS B/VB

If .A has carriage control characters, they will appear in .B as part of the logical record.

For more examples see the FILE Example Sheet No 10

FILE
2

PHOENIX COMMAND SYNTAX

	Key	Valuetype	Default	Default Qualifier	Note
(a) FILE	FROM	dsd	%C	/R	
	(TO[5] / AS[5])	dsd	-	/N	mandatory item
	OPT	text	-	-	see OPTIONS below
(b) FILE	-	dsd	-	-	mandatory item, must come first (specifies destination)
	FROM[5]	dsd	%C	/R	
	OPT	text	-	-	see OPTIONS below
both formats	STORE(integer / integer K)		32K	-	

PHOENIX NOTE

If the DCB characteristics are not defined for the destination dataset, they are taken as the Phoenix default (VBL).

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
5 Key may not be omitted.
Keyed items may occur in any order, unkeyed items in the order given.

FILE
3

JCL EXAMPLES

(a)
```
// EXEC FILE,OPT=E
//TO DD DSN=ABC1.PDS(MEM),DISP=(,CATLG),SPACE=(TRK,(2,1,5)),
//    UNIT=DISC,DCB=VB
//FROM DD *
  .
 deck
  .
//*
```

files a deck of cards as a member of a pds interpreting escaped characters.

(b)
```
// EXEC FILE
//TO DD DSN=&A,DISP=(,PASS),SPACE=(TRK,(1,1)),
//    DCB=VB,UNIT=DISC
//FROM DD DSN=ABC1.FILE(MEM2),DISP=SHR
```

copies a member of a partitioned dataset to a temporary sequential dataset.

For more examples see the File Example Sheet No. 10

JCL PARAMETERS AND DDNAMES

The input is taken from the dataset described in a dd statement named FROM.

In JCL, the output is sent to the dataset described by a dd statement named TO. If any of the DCB parameters RECFM, LRECL, BLKSIZE are not specified, they are taken to be the same as the corresponding parameters for the input. DCB parameters should always be specified when creating a disc file from card input. Options are specified using the symbolic parameter OPT.

The region size may be increased using the REG1 symbolic parameter. The default value is 32K; this is sufficient for most datasets (i.e. not VBS) with blocksize less than 2500 bytes.

For a full list of symbolic parameters and ddnames print the file INFO.JCL.PARMS(FILE).

NOTE

In all cases records will be truncated if too long for the output dataset. All records except VBS format ones must be less than 32756 bytes, and if the record is longer than this the margins option must not be used.

OPTIONS

Optional facilities are requested using the parameter OPT. They should be separated by spaces or commas. The options may be abbreviated to their initial letters.

MARGINS(n,m) Sets a 'window' on the input record. This can be used to remove sequence numbers or carriage control characters. If the actual record length is less than m, it will be used instead. The default values are (1,unset).

VERIFY This option causes the program to produce a message giving record formats, record lengths and blocksizes of the input and output, and a list of the options requested.

ESCAPE This option causes the program to recognise '@' as an escape character. The permitted escaped characters and their translations are listed in Appendix A. All others are translated into '?', but this should not be relied on. Note that whenever @L is found, the complete record is deleted.

SEQUENCE(name,n,m)
If the output record format is fixed-length records, SEQUENCE causes numbers to be placed in columns 73-80. An optional identification consisting of any alphanumeric characters may precede the numbers. It it can be from 1 to 8 characters long. SEQUENCE may be given with no arguments, with a name only, with n and m only (preceded by a comma) or with all three arguments. n and m are the starting sequence numbers and the increment, respectvely. The default values are (,100,100).

ASIS If this option is specified records are copied with no truncation of trailing spaces.

CHAR This option forces character (or segment) processing of records. This option should be used for spanned records.

RETURN CODES

A return code of 16 is issued

(a) if no SYSPRINT is provided.

(b) if any of FROM, TO or SYSPRINT do not exist.

(c) if any of FROM, TO or SYSPRINT cannot be opened.

(d) if any of FROM, TO or SYSPRINT cannot be closed.

FILEPDS

FILEPDS is a Phoenix command and JCL catalogued procedure which is designed to copy files containing one or more subfiles. These files may be partitioned files in which case the subfiles are members of the pds or sequential files in which case subfiles are a series of member-texts separated by cards defining member-names (both main and optionally alias names). FILEPDS can only handle files whose content is textual (i.e. not load modules) and as a check it will not process U format files.

Additional features allow the user a choice of sequential separators, the selection or exclusion of defined members or member-names, and also allows the application of margins and/or sequence numbers to the transferred member-texts.

Carriage control characters are handled in the style of EDIT, i.e. they are not treated as part of the text of the transferred record. Further features are provided to allow users to define their own separator cards and/or to process the transferred member-texts by the provision of a library of suitable routines.

PHOENIX EXAMPLES

(a) FILEPDS .A TO .B

The pds .A is copied to a pds .B; all members are moved.

(b) FILEPDS .A TO .B/PS OPT 'TO SEQ'

This copies the pds .A to a sequential file .B, moving all members, There are no separators between the member-texts. This is equivalent to the use of the POPS utility.

(c) FILEPDS .B TO .C OPT 'FROM SEQ,UPDTE'

The sequential dataset .B is copied to a pds .C, moving all members. The sequential dataset has IEBUPDTE format separators. This is equivalent to the use of the IEBUPDTE utility.

(d) FILEPDS .B TO .Z/PS WITH %H% LIB .MYLIB
TO SEQ,USER;FROM PDS
MARGINS 1,40;VERIFY MEMBERS
SEQUENCE WXYZ,10,10,RESET
USEREDIT
INCLUDE P1,P2,P3
INCLNAME X1
!

This transfers members P1,P2,P3 and member-name X1 with its associated member-text from a pds .B to a sequential file .Z. User defined separators are used. Margins and sequence numbers are applied to the transferred member-texts as is a user supplied processing module. All member-texts transferred are detailed.

FILEPDS
2

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
FROM	dsd	-	/R	mandatory item
TO[5]	dsd	-	/N/PO	mandatory item
WITH	dsd	%N	/R	commands
LIB	dsd	-	/R	userlib
VER[5]	dsd	*	/N	-
OPT	text	-	-	commands
STORE	(integer / integer K)	-	-	-

JCL EXAMPLES

(a)
```
// EXEC FILEPDS
//FROM DD DSN=SPQR.A,DISP=SHR
//TO DD DSN=SPQR.B,DISP=(,CATLG),UNIT=DISC,
//  SPACE=(TRK,(5,1,5)),DCB=VB
```

The pds SPQR.A is copied to the pds SPQR.B, moving all members.

(b)
```
// EXEC FILEPDS,OPT='TO SEQ'
//FROM DD DSN=SPQR.A,DISP=SHR
//TO DD DSN=SPQR.B,DISP=(,CATLG),UNIT=DISC,
//   SPACE=(TRK,(3,1)),DCB=VB
```

The pds SPQR.A is copied to the sequential dataset SPQR.B, moving all members. There are no separators between the member-texts.

(c)
```
// EXEC FILEPDS,OPT='FROM SEQ,UPDTE'
//FROM DD DSN=SPQR.B,DISP=SHR
//TO DD DSN=SPQR.C,DISP=(,CATLG),UNIT=DISC,
//  SPACE=(TRK,(5,1,5)),DCB=VB
```

The sequential dataset SPQR.B is copied to the pds SPQR.C, moving all members. The sequential file has IEBUPDTE separators. This is equivalent to the use of the IEBUPDTE program.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

FILEPDS
3

JCL PARAMETERS AND DDNAMES

The input dataset is defined by the ddname FROM, and the output dataset by the ddname TO. Commands may be passed to the program either by the symbolic parameter OPT or by the ddname WITH.

COMMANDS

Commands consist of a keyword followed, where necessary, by a series of names or numbers separated by commas. Commands are terminated by a semi-colon or a newline.

The default value is AB,INCLUDE *

ALPHABETIC or AB

This command applies only to transfers from partitioned datasets and causes the members to be transferred in the alphabetic order of main-names. This is the default ordering.

EXCLNAME or EN or XN

This command is used to define the member-names to be excluded from the transfer. The command may be followed by a list of member-names or * which means all member-names.

EXCLUDE or E or X

This command is used to define the members to be excluded from the transfer. The command may be followed by a list of member-names or * whih means all member-names.

FROM

This command defines the type of the input file. If the dataset referred to is a sequential file the statement can also define the type of searators to be used, and in some cases text to be included in the separators.

```
     FROM)   (PDS
             (SEQ[,seq-sep,seq-ext]]

seq-sep := UPDTE¦IEBUPDTE¦LISTUPDTE¦PDP¦NAG¦NAG6¦USER
seq-ext := sequential extension text for use in PDP and NAG6
           separators.
```

UPDTE, IEBUPDTE and LISTUPDTE - use the format of separators used by the IEBUPDTE utility. LISTUPDTE adds 'LIST=ALL' to the member card. This is the only form of standard separator which has a defined alias card and so should be used in most cases where membername information should not be lost in the transfer.

PDP - uses the separators that are defined for use with the PDP11/45's LINKUP program. The file title is the membername (truncated to 6 characters) and the extension is .PDS unless a seq-ext is defined in which case its text is used.

FILEPDS
4

NAG and NAG6 - -eparators used by NAG to transfer master files. This separator format can be used in cases where no aliases exist or where the text contains UPDTE format cards.

USER - separators defined by a set of routines loaded from the defined library with membernames SEPIN or SEPOUT. Further details can be obtained from the programming adviser.

If a separator type is defined then the text "SEQ," may be omitted i.e. TO SEQ,UPDTE => TO UPDTE

The default is PDS.

IGNORECC

If both input and output files contain carriage control characters then the output file will be forced to single line spacing in all cases.

INCLNAME OR IN

This command is used to define the member-names to be included in the transfer. The command may be followed by a list of member-names or by * which means all member names.

INCLUDE or I

This command is used to define the members to be included in the transfer. The command may be followed by a list of member-names or by * which means all member-names.

M[ARGINS] <n1>,<n2>

All transferred member-texts are to be margined as defined by the two numbers. The default for n1 is 1 and for n2 the record length of the output file.

NOECHO

NOECHO stops the echoing of all commands. The default is to echo commands from the WITH stream unless command input and verification are both to terminal in which case NOECHO is the default.

S[EQUENCE] ASCARD
or S[EQUENCE] <text>,<n1>,<n2>[,RESET][,NOTRIM]

Apply sequence numbers to the transferred member-texts. The ASCARD option causes the form of this number to be defined by the member number card within the sequential input file.

The alternative form defines the form of the field in the style of FILE; the RESET option causing this field to be reset at the start of each member-text and the NOTRIM option prevents the removal of the leading zeros from the numeric portion of the field.

SORTTTR or STTR

This command applies only to transfers from partitioned datasets and causes the members to be transferred in the TTR order of the members. The number of member-names that can be transferred in a given region size is increased by the use of this mode rather than alphabetic.

FILEPDS
5

TO

This command defines the type of output file. Its format is identical to the FROM command.

USEREDIT

This provides facility for a user routine to process all member-texts during transfer by loading a module (named USEREDIT) and applying a routine to all member-texts as they are transferred. Further details can be obtained from the programming adviser.

```
          (  ALL
V[ERIFY]  (  MEMBERS
          (  OPTIONS
```

OPTIONS prints out an outline of the selected options prior to starting any transfers. MEMBERS prints out details associated with the member-text after its transfer. ALL is equivalent to MEMBERS plus OPTIONS

NOTES

(a) The same member-text selected by two member-names will cause only one member to be transferred but with two member-names.

(b) Selection of members from a sequential input file MUST be selected by main-name only.

(c) Member-name - either the main name or one of the alias names.
Member-text - the text in a member.
Member - the combination of member-text and all member-names.

Main-name - all members have a main name; this will be the main-name in the XPDS sense it this exists. If it does not exist or has been explicitly excluded it will be selected from the remaining member-names.

(d) The general form of a sequential file is:-

```
file header cards
  a series of members
file trailer cards
```

and a member consists of:-

```
member header card(s)
member number card(s)
  member text
member trailer card(s)
member alias card(s)
```

For input the minimum selection of cards required is the member header cards, while for the output no cards need be defined.

FILEPDS
6

RETURN_CODES

FILEPDS issues the following return codes:-

0 - Transfer successful.
4 - Member name(s) could not be found
Lines truncated.
8 - Illegal sequential file format
Syntax error in commands.
12 - Input or output file not present
Region size too small
16 - No verification dataset.

FINISH

FINISH is a Phoenix command which terminates a console session or Phoenix job step (this is normally end of job). LOGOFF is a synonym for FINISH, see Section 4.10.6 for details.

EXAMPLE

```
FINISH
```

FORM

The SET FORM Phoenix command may be used to preset dataset formats. For resetting the default format see FD.

EXAMPLES

(a) SET F2 FB 80 1200

sets the qualifier F2 to be fixed blocked format with record length 80 and blocksize 1200.

(b) SET F1 VB 136 1228

sets the qualifier F1 to be variable blocked format with record length 136 and blocksize 1228.

(c) SET F1 VBS X 2452

This sets the qualifier F1 to be variable blocked spanned with record length greater than 32K and blocksize 2452.

Datasets with these formats may then be created by e.g.

FILE .A/F1

or INPUT .B/F2

PHOENIX COMMAND SYNTAX

SET $\begin{pmatrix}\text{FORM}\\ \text{F}\end{pmatrix}$ n string p q

n is an integer $0 \leq n \leq 9$ which defines the format number.

string the record format, is an OS record format, i.e. one of V, VB, F, FB, VBA, FA, FBA, etc. The shortened forms of DCB parameters may not be used here.

p is an integer defining the logical record length, or X allowing for line lengths greater than 32K. Note that many utilities cannot deal with LRECL=X.

q is an integer defining the blocksize.

FORTRAN

FORTRAN is a high level scientific programming language. There is a choice of the five compilers WATFIV, IBM FORTRAN IV G, G1, H and H-extended each of which may be accessed using either Phoenix or JCL procedures In addition to the procedures there is a FORTRAN command which is used mainly in connection with Fortran programming courses and which is discussed no further in this manual.

COMPILERS

G	IBM debugging compiler. Small and fast.
G1	IBM Program Product debugging compiler. To be used in preference to G.
H	IBM Optimising Compiler. Needs very large region. (192K).
H-extended	IBM Program Product Optimising Compiler. Contains features not found in H and should be used in preference to H. Also needs large region (192K).
WATFIV	University of Waterloo teaching compiler. Better error diagnostics than any or the above. It should be used for debugging if at all possible.

All these compilers accept almost the language defined in the IBM publication 'System/360 and System/370 FORTRAN IV Language' (GC28-6515). The language contains as a subset American National Standard Institute (ANSI) Fortran. G1, Hext and WATFIV each accept different extensions and modifications of the basic language.

The WATFIV compiler has the best debugging facilities of any of the compilers It has a large number of useful language extensions and data management facilities Its drawbacks are that it is a load-and-go compiler which therefore requires more store to compile and run than the other compilers. Also not all the subroutine libraries may be used with it For further details see Example Sheet No.1, the 'WATFIV User's Guide' and the 'CAMLIB Subroutine Library Specifications'.

The G and G1 compilers are the IBM debugging compilers. G1 should be used when it is inappropriate to use WATFIV. For their use see Example Sheet No 2

It should also be noted that WATFIV compiles a program three times faster than G1, but that the G1 compiled program runs three times faster than the WATFIV one

H and H-extended are optimising compilers; they produce good code but need a large amount of core and time and should be used only to produce a final load module after debugging has been completed with the G1 or WATFIV compiler

FORTRAN
2

Further details of the IBM compilers may be found in the publications:-

'Fortran IV (G and H) Programmers Guide'	(SC28-6817)
'Fortran IV (Code and Go and G1) Programmers Guide'	(SC28-6852)
'Fortran IV (H-extended) Programmers Guide'	(SC28-6865)

and further details of program development methods using these compilers may be found in Example Sheet 4.

COPYRIGHT

The G1 and H-Extended compilers, and associated libraries, are designated proprietary software, and users are referred to the Introduction to Part II of this manual

PROCEDURES

The following are the JCL catalogued procedures and Phoenix procedures:-

For WATFIV:	WATFIV, WATFIV1 (WATFIV1 is JCL only)
For FORTRAN G:	FORTGC, FORTGCL, FORTGCLG, FORTGLG
For FORTRAN H:	FORTHC, FORTHCL, FORTHCLG, FORTHLG
For FORTRAN G1:	FTG1C, FTG1CL, FTG1CLG, FTG1LG, FTG1DCLG (for go-step error tracing)
For FORTRAN H-Extended:	FORTXC, FORTXCL, FORTXCLG, FORTXLG, FORTXG

The Phoenix procedures are not available directly at a console.

PHOENIX EXAMPLES

(a) WATFIV PROGRAM=SPQR.FORTRAN

A Fortran program stored in the disc dataset SPQR.FORT (which has VB format) is compiled and run by the WATFIV load and go compiler.

(b) FTG1CLG PROGRAM=SPQR.FORT LISTC=SOURCE

The same program, but in FB format is compiled by the G1 compiler, link edited and run. The LISTC symbolic parameter is set to SOURCE in order to produce a source listing from the compiler. To use the other IBM compilers FORTGCLG, FORTXCLG or FORTHCLG should replace FTG1CLG.

(c) FTG1CLG (PROGRAM %H#) LISTC=SOURCE

```
          COMMON
            .  .
          STOP
          END
#
```

The program is supplied as a %H dataset which is defined within the environment of the procedure call. Note that in example (b) the name PROGRAM is a symbolic parameter while in (c) it is a ddname.

FORTRAN
3

(d) FORTXCL PROGRAM=.A(READ) NAMEL=.LIB(READ)

The H-extended compiler is used to produce a load module in the library .LIB from a source program contained in the member READ of the partitioned dataset .A.

(e) FTG1CLG (PROGRAM=%H# FT08F001=*A
FT09F001=.DATA/R) LIBRARY=SYS2.CAMLIB+.LIB

main program

#

A main program is compiled, link edited and run with external references resolved by automatic library scan of the three datasets SYS1 FORTLIB (built into the procedure), and SYS2.CAMLIB and .LIB which are supplied using the LIBRARY symbolic parameter. Extra data definitions for units 8 and 9 are supplied in the environment.

Further examples may be found in Example Sheets 1, 2 and 3.

PHOENIX PARAMETERS AND DDNAMES

Complete lists of all the symbolic parameters that may be used with the Phoenix procedures may be produced by using the command 'HELP PROC procedure-name'.

For all the procedures that include a compilation step, the source program must be given using either the symbolic parameter or the ddname PROGRAM In the procedures that invoke the IBM compilers, the most useful symbolic parameter is LISTC (default value NOSOURCE) which may be set to SOURCE to get a source listing. An abbreviated compilation output may be obtained from the G1 and H-extended compilers by setting PRINTC (default *) to $/FN, BRIEFC (default $) to * and LISTC to TERM,NOSOURCE

In any link edit step, the LIBRARY symbolic parameter is used to supply extra load module libraries for the automatic library scan mechanism. The symbolic parameter or ddname LKIN can be used for linkage editor control statements while NAMEL (default value &GOSET:MAIN) defines the partitioned dataset used for the load module output from the linkage editor. Library scan is controlled by the parameter CAL while listing options may be changed using LISTL, MAPL or XREFL which have default values of LIST, NOMAP and NOXREF respectively.

In the go steps of the IBM compiler procedures, the dataset for unit 5 (e.g. for READ(5,... statements) may be defined by symbolic parameter FT05 or ddname FT05F001 while unit 6 (e.g. for WRITE(6,... statements) may be changed using the parameter FT06. Other units are set up using the ddnames FTnnF001 where nn is 01-04 and 07-16.

FORTRAN
4

The WATFIV procedure has some differing parameters, for example, the cpu time for the complete procedure is controlled by parameter SECG (with default 0 which is equivalent to no time limit) although it should be noted that the compiler itself has a built-in limit of 5 secs. Listing and other compiler options are changed by using the parameter LISTG while the dataset for unit 5 is defined using either the symbolic parameter or the ddname DATAFILE.

JCL EXAMPLES

(a)
```
// EXEC WATFIV
//PROGRAM DD DSN=SPQR.FORTRAN,DISP=SHR
```

A Fortran program stored in the VB disc dataset SPQR.FORT is compiled and run by the WATFIV load-and-go compiler.

(b)
```
// EXEC FTG1CLG,LISTC=SOURCE
//FORT.SYSIN DD DSN=SPQR.FORT,DISP=SHR
```

The same program,but in FB format, is compiled, link edited and run using G1. The source listing is requested using the symbolic parameter LISTC.

(c)
```
// EXEC FTG1CLG,LISTC=SOURCE
//FORT.SYSIN DD *
        COMMON

        STOP
        END
//*
```

The program is supplied as an instream dataset.

(d)
```
// EXEC FORTXCL,NAMEL='SPQR.LIB(READ)',DISPL=OLD
//FORT.SYSIN DD DSN=SPQR.A(READ),DISP=SHR
```

The H-extended compiler is used to produce a load module in the library .LIB from a source program contained in the member READ of the pds SPQR.A

(e)
```
// EXEC FTG1CLG,REGG=100K
//FORT SYSIN DD *
        main program

//*
//LKED SYSLIB DD
//            DD   DSN=SYS2.CAMLIB,DISP=SHR
//            DD   DSN=SPQR LIB,DISP=SHR
//GO.FT08F001 DD   SYSOUT=A
//GO FT09F001 DD   DSN=SPQR.DATA,DISP=SHR
```

FORTRAN
5

A main program is compiled, link edited and run with external references resolved by automatic library scan of the three datasets SYS1.FORTLIB (built into the procedure), and SYS2.CAMLIB and .LIB which are supplied with the concatenated SYSLIB dd statements. Extra dd statements for units 8 and 9 are supplied. The region size for the GO step is raised from the default of 48K to 100K.

JCL SYMBOLIC PARAMETERS AND DDNAMES

Complete lists of all the symbolic parameters that may be used with the JCL catalogued procedures may be found in the partitioned dataset INFO.JCL PARMS in members corresponding to procedure names.

The source program must be supplied to all the compilers by using the ddname SYSIN (or PROGRAM for WATFIV only). Source listings may be obtained from the IBM compilers by using the symbolic parameter LISTC (default NOSOURCE) with value SOURCE and from WATFIV by using the parameter LISTG with value LIST. LISTG is also used to change default time and debugging options.

Extra libraries are supplied to the linkage editor by concatenation on the SYSLIB dd statement, and control statements, may be given by using the SYSIN dd statement. The linkage editor load module output dataset may be defined by using the NAMEL and DISPL parameters or by overriding the SYSLMOD dd statement Listing options are controlled by the parameters LISTL (default LIST), MAPL (default NOMAP) and XREFL (default NOXREF) while the automatic library scan default of CALL for -CLG and -LG procedures and NCAL for -CL procedures may be changed by using the CAL parameter.

In the GO steps for the IBM compilers, the datasets for units 5, 6 and 7 may be defined using symbolic parameter FT05, FT06 and FT07 or by overriding dd statements FT05F001, FT06F001 and FT07F001, while the other units up to and including unit 16 may be used by supplying additional dd statements FT[illegible]F001. The region size for the GO steps may be changed from the default of 48K by setting the parameter REGG. In WATFIV, data for unit 5 may be supplied using ddname DATA.

The WATFIV1 procedure is a more basic version of the WATFIV one. If WATFIV1 is used then a dataset defined by the ddname SYSIN or FT05F001 must be supplied containing not only the source program but also the WATFIV control statements $JOB and $ENTRY, which, if the WATFIV procedure is used, are supplied automatically.

SUBROUTINE LIBRARIES

There are several subroutine libraries available for the Fortran user, containing routines covering numerical, statistical and graphical applications as well as input/output and logical operations.

FORTRAN
6

The libraries for the IBM compilers are partitioned datasets containing load modules which may be accessed with the linkage editor via the automatic library call mechanism or via explicit INCLUDE statements. The former method is recommended and is achieved by ensuring the library is included on a SYSLIB data definition statement.

WATFIV cannot process load modules and so the libraries are kept in source or object module form.

Full descriptions of all routines in the libraries may be found by reference to the manual 'Summary of Applications Software'.

The Numerical Algorithms Group (NAG) Library should if applicable be used in preference to the other libraries.

Library	Compilers	PDS name	Built into procedures	Notes
Built in functions	IBM	SYS1.FORTLIB	Yes	Contains trig fns, I/O routines and error handling
"	WATFIV	S.WATFIV.FUNLIB S.WATFIV.WATLIB	Yes	"
Numerical Algorithms Library (NAG)	IBM	SYS2.NAGLIB	No	Numerical and statistical routines
"	WATFIV	SYS2.WATFIV.NAGLIB	Yes (Not in WATFIV1)	"
IBM Scientific Subroutine Package	IBM	SYS2.SSPLIB	No	"
Harwell Library	IBM	SYS2 FORTLIB	No	"
Cambridge	IBM	SYS2.CAMLIB	No	Graphical, logical and I/O routines
"	WATFIV	SYS2.WATFIV.CAMLIB	Yes (Not in WATFIV1)	"

FORTRAN
7

Not all the Cambridge routines are available under WATFIV. For details see 'CAMLIB Subroutine Library Specifications'.

Copies of source or load modules may not be taken unless permission has been granted by the Consultancy Service. This is because of copyright restrictions.

NOTES

All the compilers (except WATFIV) require the source program to be in FB format. See Example Sheet 4.

When intending to read within a Fortran program from a member of a partitioned dataset, LABEL=(,,,IN) must be coded, otherwise an end-of-dataset error will occur as soon as reading starts.

Only units 1 to 16 inclusive may be used for I/O operations in a Fortran program.

ERROR MESSAGES

WATFIV error messages are self-explanatory and are described in the 'WATFIV User's Guide'. Error messages from the G and H compilers may be found described in the corresponding Programmers Guide while error messages from the G1 and H-extended compilers are to be found in the publications :

'Fortran IV (H-extended) Compiler and Library (Mod II) messages' (SC28-6865)
and 'Fortran IV(G1) Processor and TSO Fortran Prompter Installation Reference Manual' (SC28-6856)

Further and more detailed information on the handling of Fortran errors will be found in the manual 'Error Tracing and Debugging Aids'.

HELP

HELP is a Phoenix command and JCL catalogued procedure which provides information about many aspects of the system.

PHOENIX EXAMPLES

(a) HELP

gives information on how to use HELP.

(b) HELP FILE

types many lines of information about files and the FILE command.

(c) HELP FILE TITLES

types a few lines about dataset names.

(d) HELP FILE TITLES PDS

produces a description of file titles and of partitioned datasets.

(e) HELP STATUS

types details of the STATUS command.

(f) HELP USER BLOGGS SMITH

gives the userids and departments of users BLOGGS and SMITH.

PHOENIX COMMAND SYNTAX

HELP [TO[5] dsd][1] major-keyword[1] qualifying-keywords[1]

JCL EXAMPLES

(a) // EXEC HELP

prints information on how to use HELP

(b) // EXEC HELP,KEY=DOCUMENT

prints a list of the main manuals available, together with their amendments.

JCL PARAMETERS AND DDNAMES

The HELP keyword, if present, is defined by the symbolic parameter KEY. Output is defined by the symbolic parameter or ddname TO, and is to the printer, by default.

KEYWORDS

HELP is a valid command on its own or it may be followed by one or more keywords which describe the help needed. For a list of currently available major keywords run the XPDS program on the file INFO.HELP.

---

1 Optional
5 Keyword may not be omitted

HELPK

HELPK is a Phoenix command which may be used to access private HELP files or to save the output from private or the standard HELP files on disc. For a full specification of the HELP program see the dataset INFO.SPEC.HELP.

PHOENIX EXAMPLES

(a) HELPK TO &TEMP FILE OPTIONS

This gives information about the options available for use with the file command. Output is stored in a temporary dataset &TEMP.

(b) HELPK FROM .MYHELP SUBP1 SUBP2

A private help file is used to give information about SUBP1 and SUBP2.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
TO[5]	dsd	*	/R	-
FROM[5]	dsd	INFO.HELP	/N	-
-	text	-	-	keywords

NOTES

For further details of the HELP and HELPK command see HELP and the dataset INFO.SPEC.HELP.

HP

HP is a Phoenix command which invokes a program to simulate an HP35 calculator. For a brief specification of the program type HELP HP at a console.

IBMCMPRS

IBMCMPRS is a JCL catalogued procedure which condenses a partitioned dataset and copies it back to a pds of the original name. IBMCMPRS uses the IBM utility IEBCOPY, and should only be used if the Cambridge utility COMPRESS cannot be used.

JCL EXAMPLES

(a) `// EXEC IBMCMPRS,NAME='SPQR.LIB'`

This compresses the partitioned dataset SPQR.LIB.

(b)
```
// EXEC IBMCMPRS,NAME='SPQR.LINKLIB',DSN='SPQR.TEMP',
//    SPACE='(TRK,(5,1,5))'
```

This compresses the pds SPQR.LINKLIB using SPQR.TEMP as a safety dataset. The size of the datasets is increased from the default value.

JCL PARAMETERS AND DDNAMES

The pds to be compressed is defined by the symbolic parameter NAME. A dataset to be used as a backup copy, in case of failure, may be specified by the symbolic parameter DSN. If the dataset is large, its size may be defined by the SPACE symbolic parameter. The default size for the SPACE parameter is 1,1,5.

NOTES

IBMCMPRS condenses the dataset to a temporary dataset, deletes the original dataset, then condenses the temporary dataset to a permanent dataset with the same name as the original file. If the dataset contains valuable information a permanent dataset should be used instead of the temporary one.

IBMCOPY

IBMCOPY is a Phoenix command and JCL catalogued procedure which uses the IBM IEBCOPY utility for moving and merging partitioned datasets. The Phoenix command is not available at a console. The Cambridge utility CONDENSE is recommended in preference to IBMCOPY.

PHOENIX EXAMPLES

(a) IBMCOPY FROM .AFILE TO .BFILE

condenses .AFILE to .BFILE.

(b) IBMCOPY .CFILE TO .DLIB WITH &A

copies .CFILE to .DLIB taking the control statements from a temporary dataset &A which must be of format FB.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
FROM	dsd	-	/R	mandatory item
TO	dsd	-	/N	mandatory item
WITH[5]	dsd	SYS2.CONTROLS:CNDNSE2	/R	
WORKSPCE[5]	text	MED	-	
VER	dsd	*	/N	
STORE	(integer / integer K)	64K	-	

JCL EXAMPLES

(a)
```
// EXEC IBMCOPY
//TO DD DSN=ZYX8.ZLIB,DISP=(,CATLG),SPACE=(TRK,(5,1,3)),
// UNIT=DISC,DCB=VB
//FROM DD DSN=ZYX8.YLIB,DISP=SHR
```

condenses from ZYX8.ZLIB to ZYX8.YLIB using default options.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

(b)
```
// EXEC IBMCOPY
//TO DD DSN=ABC1.LIB,DISP=OLD
//FROM DD DSN=ABC1.NEWLIB,DISP=SHR
//SYSIN DD *

  IEBCOPY commands

//*
```

This merges ABC1.NEWLIB into ABC1.LIB obeying commands in the instream dataset.

(c)
```
// EXEC IBMCOPY,CNTRLS='SPQR.CNTLS'
//TO DD DSN=SPQR.LINKLIB,DISP=OLD
//FROM DD DSN=SPQR.TESTLIB,DISP=SHR
```

SPQR.TESTLIB is merged with SPQR.LINKLIB. Control statements are passed to the program by the CNTRLS symbolic parameter which defines the SPQR.CNTLS dataset.

JCL PARAMETERS AND DDNAMES

The input pds is defined by a dd statement with ddname FROM. The output pds, which may already exist, should be defined by ddname TO. Control statements may be passed to the program by a statement with ddname SYSIN or by setting the symbolic parameter CNTRL1 to null and CNTRLN1 to the name of a dataset containing control statements; the format of this dataset must be FB.

CONTROL STATEMENTS

For details of the control statements see the IBM 'Utilities Manual' - GC28-6586.

SYS2.CONTROLS(CNDNSE2) contains the control statement

```
COPY INDD=FROM,OUTDD=TO,LIST=NO
```

and is the default for both the JCL procedure and the Phoenix command.

IF

IF is a Phoenix control command which may be used for making tests in a command sequence.

PHOENIX EXAMPLES

(a) IF BADFILE QUIT

If the BADFILE flag is set control returns to the next level of command.

(b) IF %G5 GT 10 JUMP ON

If the variable %G5 is greater than 10 control is transferred to the label ON.

(c) IF SW1 SET %L1=0

If SW1 is set %L1 is set to 0.

(d) IF -ERROR SET SW1

If the ERROR switch is not set, user switch 1 is set.

(e) IF PASSWORD IS SET SUBMIT .JOB

If the parameter PASSWORD has a value the job in .JOB is submitted.

(f) IF ROUTE EQ NEWCASTLE AND %USER EQ FRED PARS ROUTE=RECEPTION

If parameter ROUTE has the value NEWCASTLE and the parameter %USER has the value FRED then ROUTE is reset to RECEPTION.

(g) IF .T:G,H,J AVAILABLE COMMENT Members G,H,J exist

PHOENIX COMMAND SYNTAX

IF condition command

a condition is an optional $\underset{\text{NOT}}{\pm}$ followed by

switch-name
flag-name
arithmetic-variable relation value
parameter-name IS SET
parameter-name IS UNSET
dataset-name AVAILABLE
dataset-name IS AVAILABLE
parameter-name relation text
parameter-name relation 'quoted text'

The command should not refer to a here dataset.

IF
2

NOTE

See section 4.10.5 for further details of the IF command.

IMAGE

IMAGE is a Phoenix command and JCL catalogued procedure that provides line-imaging for sequential datasets with optional interpretation of escape combinations. All record formats except VBX may be used.

PHOENIX EXAMPLES

(a) IMAGE OPT S

This command takes the current document, assumed to be a dataset originally produced using the stream paper tape input option, and produces an output document line-imaged to reconstruct the original record layout of the paper tape.

(b) IMAGE TO .IMAGED.FILE FROM .BINARY.FILE OPT=F,NE

A dataset .BINARY.FILE created by binary input from a 7 track paper tape in Titan Flexowriter code is processed by IMAGE to produce a line-imaged dataset with no interpretation of the escape character.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Note
FROM	dsd	%C	/R	-
TO	dsd	%O	/N	-
OPT	text	-	-	see OPTIONS below
STORE	(integer / integer K)	20K	-	-
VER	dsd	*	-	-

Choose one of the alternatives in brackets
Any key or item may be omitted unless otherwise stated
Keyed items may occur in any order; unkeyed items in the order given.

JCL EXAMPLES

```
// EXEC IMAGE,OPT='F,NE'
//TO DD DSN=PS18 IMAGED FILE,DISP=(,CATLG),
//  UNIT=DISC,DCB=VB,SPACE=(TRK,(3,3,1))
//FROM DD DSN=SPQR BINARY.FILE,DISP=SHR
```

A dataset created by reading a binary paper tape, which was punched on a 7 track flexowriter, is imaged.

JCL PARAMETERS AND DDNAMES

The input is taken from the dataset defined with a dd statement named FROM

The output is sent to the dataset defined with the dd statement TO. If any of the DCB parameters RECFM, LRECL or BLKSIZE is not specified it is taken to be the same as the corresponding parameters for the input. If the RECFM for the output dataset specifies carriage control characters, then they are set in each record of the output dataset to 'advance one line' unless changed by an escape combination (see below). Error messages are sent to SYSPRINT or to the HASP log if SYSPRINT is missing

Options may be specified using the symbolic parameter OPT

The region size may be increased using the symbolic parameter REGl. The default value is 20K; this is sufficient for datasets with blocksize less than 2500 bytes

For a full list of symbolic parameters and their defaults list the file INFO JCL PARMS(IMAGE)

OPTIONS

Options are given using the parameter OPT They may be separated by spaces or commas and may occur in any order

F, 7 or FLEXO	Input is in 7 track flexowriter code
A, 8 or ASCII	Input is in 8 track ASCII code
C or CARDS	Input is in EBCDIC (default)
S or STREAM	Input is from paper tape using STREAM mode
E or ESC	Switch on interpretation of escape characters
NE or NOESC	Switch off interpretation of escape characters
N or NOIMAGE	Switch off line imaging
T or TABSET	Set tab positions

If A, S or F is set, E is also set

STREAM mode is fully defined in the paper tape input specification, (see Section 3 4.5), but briefly, STREAM mode is used to reconstruct the record structure of a paper tape dataset since on reading paper tape the records are divided at 80-byte intervals.

IMAGE
3

CARRIAGE CONTROL CHARACTERS

The sole function of carriage control characters (newline, carriage return, linefeed and formfeed) in the input from a binary paper tape source is to terminate the line. No interpretation of the particular character used is implemented When reading ASCII code, multiple carriage returns are treated as one carriage return, and a carriage return (or group of carriage returns) immediately adjacent to a linefeed (or formfeed) is ignored. Formfeed is treated as linefeed.

LINE IMAGING

On a device such as a flexowriter which has a backspace key, there are many ways of producing the same line of type. For example the key sequences 'ABCD' and 'A<space>C<backspace><backspace>B<space>D'. The object in line-imaging input from such a device is to produce inside the computer a record whose contents are independent of the particular sequence of keys struck. The final form of the record can therefore be deduced by looking only at the Flexowriter typescript. In practice complications arise because of the need to remove erased characters, interpret escape combinations, and nevertheless keep tabbed fields in line.

Line-imaging is performed before the interpretation of escape combinations, so that, for example, the sequence '@A<backspace> <erase>E' gives the same result as '@E'. A fault character is generated if overprinting of different characters occurs (unless one is erase, of course) When the line-image is complete, trailing spaces and erases are removed and escape combinations interpreted. Characters to the right of erases and escaped characters, up to the next character position in the line that was reached by means of a tab character, are shifted to the left The line-image buffer is 256 characters long

ESCAPE COMBINATIONS

The standard escaped characters as defined in Appendix A are supported with the exception of @I and @Q. In addition, the escape combinations @0, @1, @2, @3, @9 cause the carriage control character in the current record to be set to no line throw, one, two, three line throws and page throw if the output records contain control characters. Note that the control character acts before the line it is in for datasets with ASCII carriage controls.

TABS

Default tab settings are at every 8th position. If one of the F, 7, FLEXO or T F options is set then at least two spaces will occur before each position reached by tabbing, otherwise at least one.

IMAGE
4

Tab positions may be changed by

```
     T n1 n2 . .
or   T ASM  which is equivalent to T 10 16 36 72
or   T FORT which is equivalent to T 7 72
     T PDP  which is equivalent to T 9 17 41 73
```

Note that the T F option may be followed by ASM, FORT, PDP or explicit tab settings. e.g.

```
     T F ASM
```

ERROR HANDLING

If the line-image buffer overflows, the line so far is terminated with newline and the remainder is treated as a new line. If an output record overflows, a new record is started. Fault characters and unknown escape combinations are turned into the character X'FF'.

INFO

INFO is a Phoenix command which outputs dynamic system information. INFO followed by no keywords types a list of all possible keywords.

EXAMPLES

(a) INFO TIME

The time of day is output.

(b) INFO CPU

The cpu time used so far in the session or current job step is output.

(c) INFO TO .A DATE

The day and month are copied to the file .A.

(d) INFO TIME FIND N,WHO,'Public'

The time of day, number of users logged on, their userids and users logged on at public consoles is typed.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
TO[5]	dsd	*	/N	-
-	INFO keyword	-	-	see below

Keywords	Information Supplied
CHARGE	charge for current online session (not available offline).
CONNECT	the connect time for the current session.
CPU	the cpu time used in the console session or in the current job step (if in a job).
DATE	the day and month.
DAY	the day of the week, day of month, month and year.

Any key or item may be omitted unless otherwise stated.
5 Key may not be omitted.

INFO
2

DD	details of existing dd statements. For those created by JCL, 'JCL' is printed; for those created by Phoenix it gives the type of device (see below) and (if disc) the name of the file. It may also give one or more of the following letters: W - file is writeable (disposition NEW,OLD,MOD) O - file is currently open via this dd statement P - file has been previously open via this dd statement.
DISC	Disc I/O time only
FIND 'text'	all users logged on whose name or location contains the given text. Upper and lower case letters are synonymous. (This requires extra store.)
FIND ALL	users logged on and their locations.
FIND LINES	the location of all Phoenix lines. While every effort is made to keep this up-to-date reliability cannot be guaranteed.
FIND userid(s)	If a user is logged on, his physical location is printed, otherwise a message 'userid IS NOT LOGGED ON' is printed. (Userids must be separated by commas).
FIND N	number of logged on users.
FIND WHO	lists logged on users.
IO	Disc I/O time used in current online session or in the current job step if in a job. Tape and disc times are given separately.
ISODATE	date in ISO format, i.e. year, month, date.
JOBNUMBER	job number (only available offline).
PRIORITY	priority of job or console session.
PROJECT	project number of job or console session.
REGION	the region size and the maximum store used so far in the session and job step.
RESOURCE	number of resource units used in the current online session. (Not available offline.)
STORE	the store used in the last Phoenix command at the current level or, after popping up a level, the store used by the C command.

TEMPFILE	a list of currently existing temporary files, excluding those created by JCL.
TERMINAL	SET TERMINAL options set for this console session (not available offline).
TIME	time of day.
USER	identifier of logged on user or userid under which the job is running.

Multiple keywords may be separated by spaces, e.g. INFO CPU IO

DEVICES FOR DD_OPTIONS

DUMY	dummy
DISC	disc
INTR	internal reader
PRTA	*A or SYSOUT=A
PRTC	*C or SYSOUT=C
TERM	Phoenix console

RETURN_CODES

The following return codes are issued:

4	at least one of the users named in an INFO FIND command was not logged on.
8	syntax error, unknown keyword.
16	unable to open SYSPRINT or I/O error on SYSPRINT.

NOTE

Until online charging is implemented, CHARGE gives the same value as RESOURCE.

INIT

The Phoenix command INIT has been superseded by MKINIT.

INPUT

INPUT is a Phoenix command which reads data from a console. The resulting dataset may be a permanent or temporary dataset or the current document.

EXAMPLES

(a) INPUT
This goes to the current
document
/*

(b) INPUT &A/FB
This goes to a temporary dataset
as blocked card images
/*

(c) INPUT TO .A:A/VB OPT=M(1,72)
This goes to a member of a pds
/*

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
TO	dsd	%O	/N
OPT	-	option list	see OPTIONS below
STORE	(integer / integer K)	32K	-

OPTIONS

The OPT keyword may be followed by any of the options defined for the FILE command.

NOTES

Text should be typed immediately after the INPUT command and should be terminated by the characters /* in the first two character positions of a line immediately followed by carriage return.

INPUT is only available at a console and not within Phoenix jobs in which the CURRENT command may be used instead.

Choose one of the alternatives in brackets.
Any item may be omitted, unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

JUMP

JUMP is a Phoenix control command which causes transfer of control in a command sequence to the label specified. Instream (here) documents within the command sequence should not contain the label or else confusion may result. The label is defined by *label at the beginning of a line. If the transfer is a forward jump, SKIP should be used. This command may only be used within a command sequence. See Section 4.10.4 for further details.

EXAMPLE

```
*B
WAIT 2
MESSAGE
IF RC JUMP B
```

This command sequence will loop until a message is received.

PHOENIX COMMAND SYNTAX

```
JUMP label
```

NOTE

The search for the label is from the beginning of the current command sequence, provided the sequence is held on a rewindable medium, otherwise from the current position, skipping one or two records.

KILL

KILL is a Phoenix command which may be used to kill a job.
Jobs which are executing, being read in, printing, plotting or punching may not be killed. Jobs which have not yet started executing or which have already executed and are waiting to print, punch or plot, or are waiting to be collected may be killed. To kill a job, the user must have suitable jobqueue access (see PASSWORD).

PHOENIX EXAMPLES

(a) KILL 29

Job number 29 for the set user is killed.

(b) KILL J029 SPQR

Job number 29 for user SPQR is killed.

(c) KILL PASSWORD SECRET USER=SPQR JOB=12

Job number 12 for SPQR is killed.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Notes
JOB	text	-	-
USER	text	defaults to value of U	-
PASSWORD	text	-	-

NOTE

If a userid is not given, then the value of the Phoenix parameter 'USER' or 'U' is used.

Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

LABEL

LABEL is a Phoenix command which alters the label of a disc dataset. This label, which is also known as the Dataset Control Block (DSCB), contains information about the dataset such as the record format, blocksize and logical record length. LABEL does not make any changes to the file itself. Changing the dataset label is a potentially dangerous activity; users should therefore take care when using this program. The only output from the program is an error message in cases of failure.

PHOENIX EXAMPLES

(a) LABEL .JUNK BLKSIZE=400

The blocksize field in the label of the dataset .JUNK is changed to 400 bytes.

(b) LABEL .ABCD SECONDARY=2 TRK CONTIG, RECFM=VBA

The record format of .ABCD is changed to VBA and the secondary allocation to 2 tracks.

(c) LABEL TO .RES .DEF LRECL=40

The logical record length of .DEF is set to 40 bytes and verification sent to the dataset .RES.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
TO[5]	dsd	*	/N	Must be first
-	dsd	-	/R	Mandatory item
-	text	-	-	Mandatory item giving the data.

DATA

The data for the command consists of a series of keyword/value pairs. The keywords are the names of fields in the DSCB and are separated from their values by =. The more common field names have as synonyms the corresponding DCB names. Keyword/value pairs may be separated from each other by space or comma. Any value may be specified as a hexadecimal number starting with X' and ending with '. The number of hexadecimal digits must correspond exactly to the size of the field being set.

---

5 Keyword must not be omitted.

LABEL
2

Most values may also be specified in a conventional format appropriate to the particular field. The program does not check that the values given are at all reasonable. The fields which can be set are as follows:-

BLKSIZE or DS1BLKL length 2

The blocksize can be specified as a decimal number.

DIRCT or DS1NOBDB length 1

The number of bytes used in the last directory block of a pds can be specified as a decimal number.

DSORG or DS1DSORG length 1

The organization of the dataset can be specified as one of the words PS, PO or DA.

DS1DSIND length 1

The dataset indicators can only be specified in hexadecimal. Only the X'20' bit is permitted to be changed. Other bits are ignored.

KEYLEN or DS1KEYL length 1

The key length can be specified as a decimal number.

LRECL or DS1LRECL length 2

The logical record length can be specified as a decimal number.

LSTAR or DS1LSTAR length 3

The relative address of the last block written to the dataset can only be specified in hexadecimal.

OPTCD or DS1OPTCD length 1

Option codes can be specified as one or more of the following letters, with no spaces between them:-

W C Z T E F A R

In addition, the letter N can be used as a dummy; it causes no bits to be set.

RECFM or DS1RECFM length 1

The record format can be specified as one or more of the following letters, with no spaces between them:-

F V U T B S A M

In addition, the letter N can be used as a dummy; it causes no bits to be set.

RKP or DS1RKP length 2

The relative key position in a data block can be specified as a decimal number.

SECONDARY or DS1SCALO length 4

The secondary allocation quantity for the dataset can be specified as a decimal number followed by one of the words TRK, BLK or CYL. The word CONTIG may optionally follow the units word. One or more spaces are required as separators between the various parts of the value.

TRBAL or DS1TRBAL length 2

The number of free bytes left on the last track can be specified as a decimal number.

ERRORS

After any error the program reflects the input data with '>' underneath the last character read. This is followed by a self-explanatory error message. The program then stops with return code 12. No updating takes place. If SYSPRINT cannot be opened to output an error message, a return code of 16 is given.

LIBRARY

LIBRARY is a Phoenix command which opens a library of Phoenix commands for scanning. Any library set up by this command is scanned before the system libraries. If several libraries are set up, they are scanned in reverse order of setup until the command is found. Private libraries may be constructed by use of the MKLIB procedure (see MKLIB below) or by explicitly constructing and assembling macro commands (see Appendix E).

PHOENIX EXAMPLES

(a) LIBRARY .LINKLIB:MEM1

After this command .LINKLIB:MEM1 is scanned before the system libraries to find any subsequent commands.

(b) LIBRARY CANCEL

cancels the last library set up.

(c) LIBRARY CANCEL ALL

cancels all except system libraries.

PHOENIX COMMAND SYNTAX

To set up a library

LIBRARY dsn:member

To cancel a library

LIBRARY CANCEL ALL

LIBRARY CANCEL

NOTE

The following library is available:

NORMAL makes available the procedure library %CL

(LIBRARY NORMAL is set up at the beginning of a job).

LINKAGE EDITOR

The linkage editor is a program that produces a load module from one or more load or object modules. It is accessed via the Phoenix and JCL procedures LKED and LKEDG as well as via the large number of Phoenix and JCL language procedures that have names terminating in -CL, -CLG and -LG.

Full details of the linkage editor may be found in the IBM manual 'Linkage Editor and Loader' (GC28-6538).

PHOENIX EXAMPLES

(a) LKED LOADSET= OBJ1+.OBJ2 NAMEL= .LIB(PROG1)

The linkage editor combines the two object modules contained in the datasets OBJ1 and .OBJ2 and places the output load module named PROG1 in the partitioned dataset .LIB.

(b) LKEDG (LKIN %H# BIL= LIB INPUT=.A OUT=*) LIBRARY=.LIB
```
 INCLUDE BIL(PROG1)
 ENTRY A
#
```

Control sections are processed from the member PROG1 of the dataset LIB and the automatic library scan mechanism scans the library given by the parameter LIBRARY. The output load module is run with the data definitions with ddnames INPUT and OUT defining the data and the output for the program

Examples of the use of the linkage editor may be found in Example Sheet 20 or elsewhere in Part II of this manual for language procedures.

PHOENIX PARAMETERS AND DDNAMES

The primary input, if any, may be defined using the symbolic parameter LOADSET and the linkage editor control statements may be given using the symbolic parameter or the ddname LKIN. The load module output is defined using the symbolic parameter NAMEL (default value &GOSET:MAIN) while any libraries needed are supplied using the symbolic parameter LIBRARY or the ddname SYSLIB. Linkage editor options may be changed using the parameters LISTL (default value LIST), MAPL (default value NOMAP), XREFL (default value NOXREF) and CAL (default depends on the procedures).

More information about these and the other symbolic parameters available may be found by using the command 'HELP PROC procedure-name'.

LINKAGE EDITOR
2

JCL EXAMPLES

(a)
```
// EXEC LKED,DISPL=OLD,NAMEL='SPQR.LIB(PROG1)'
//LKED.SYSIN DD DSN=SPQR.OBJ1,DISP=SHR
//          DD DSN=SPQR.OBJ2,DISP=SHR
```

The linkage editor combines the two object modules contained in the datasets .OBJ1 and .OBJ2 and places the output load module named PROG1 in the partitioned dataset .LIB.

(b)
```
// EXEC LKEDG
//LKED.SYSLIB DD DSN=SPQR.LIB,DISP=SHR
//LKED.BIL    DD DSN=SPQR.LIB,DISP=SHR
//LKED.SYSIN  DD  *
  INCLUDE BIL(PROG1)
  ENTRY A
//*
//GO.INPUT    DD DSN=SPQR.A,DISP=SHR
//GO.OUT      DD SYSOUT=A
```

Control sections are processed from the member PROG1 of the dataset .LIB and the automatic library scan mechanism scans the library given by the dd statement SYSLIB. The output load module is run with the data definitions with ddnames INPUT and OUT defining the data and the output for the program.

JCL PARAMETERS AND DDNAMES

The primary input and any control statements may be defined either on an overriding dd statement SYSLIN or on an additional dd statement SYSIN. The load module output is defined using the symbolic parameter NAMEL or an overriding dd statement SYSLMOD; any libraries needed are given on an overriding dd statement SYSLIB.

ERRORS

Information on linkage editor errors may be found in Chapter 8 of 'Error Tracing and Debugging Aids', and in the IBM manual 'Linkage Editor and Loader' (GC28-6538).

LIST

LIST is a Phoenix command and JCL catalogued procedure which prints a dataset (which may be sequential, partitioned or member of a pds) on the line printer with line numbers in multi-column format. Options are available to exclude or include members of a pds, and to control the format of the output.

PHOENIX EXAMPLES

(a) LIST .PROG

The sequential dataset .PROG is listed on the printer with line numbers.

(b) LIST .PDS OPT 'E MEMX'

The partitioned dataset .PDS is listed on the line printer with line numbers. The member 'MEMX' is not printed.

(c) LIST .A+.B TO .C/N/VBA OPT W120

The datasets .A and .B are copied with line numbers added to the dataset .C. The maximum line width is 120 characters.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
FROM	dsd	%C	/N
OPT	text	LN	-
TO[5]	dsd	*A	/N
WITH[5]	dsd	-	/R
VER[5]	dsd	*C	/N
STORE	(integer / integer K)	-	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Key may not be omitted.

LIST
2

JCL EXAMPLES

(a) // EXEC LIST,NAME='SPQR.PROG'

The sequential dataset SPQR.PROG is listed on the printer with line numbers.

(b) // EXEC LIST,NAME='SPQR.PDS',OPT='E MEMX'

The partitioned dataset SPQR.PDS is listed on the line printer with line numbers. The member 'MEMX' is not printed.

(c)
```
// EXEC LIST,OPT=120
//FROM DD DSN=SPQR.A,DISP=SHR
//   DD DSN=SPQR.B,DISP=SHR
//TO DD SPQR.C,DISP=(,CATLG),SPACE=(TRK,(2,1)),
//   UNIT=DISC,DCB=VBA
```

The datasets SPQR.A and SPQR.B are copied with line numbers added to the dataset SPQR.C. The maximum line width is 120 characters.

JCL PARAMETERS AND DDNAMES

The input is defined by the symbolic parameter NAME or the ddname FROM. Output is defined by the symbolic parameter or ddname TO. Options are passed to the program by the symbolic parameter OPT.

OPTIONS

The default option for the list program is 'LN'. For details of the options available see PRINT.

LISTPDS

The Phoenix command and JCL catalogued procedure LISTPDS have been superseded by PRINT and LIST.

LOCAL

LOCAL is a Phoenix command which may be used to set or unset values of symbolic parameters. Values set in this way remain set only until the end of the command sequence in which they were set. They may only be passed to a called sequence by the use of + in the C command (see section 4.8.3).

PHOENIX EXAMPLES

```
LOCAL PA1=abcd PRINTER=5K -TIME
...
...
C .NEXT.LEVEL +
```

The parameter TIME is unset and the parameters PA1 and PRINTER are set at the current level and then passed to the next level by the + on the C command.

PHOENIX COMMAND SYNTAX

```
LOCAL parameter-list
```

where parameter-list is any number of items of the form

(parameter-name =[1] value
  -parameter name)

---

1 optional

LOGON

LOGON is a command which is used to start a Phoenix console session and to control the resources used by it. Details of the LOGON sequence are given in Section 2.2.4. After LOGON is typed by the system in response to a carriage return, the user should type his userid followed by carriage return. (The user should not type LOGON himself).

PHOENIX EXAMPLES

(a) LOGON SPQR

The user SPQR logs on to the system using a previously set up project number.

(b) LOGON SPQR 1234

The user SPQR logs on using project number 1234.

(c) LOGON RQPS 9999 PASSWORD SECRET

The user RQPS logs on to Phoenix quoting his project and password.

PHOENIX COMMAND SYNTAX

The syntax for the LOGON line is

LOGON userid PROJECT[1] project-number[1] [keyword[5]value][13]

KEYWORDS

PASSWORD PW	The value may have up to eight characters or digits.
PRIORITY P	The value (<256) specifies the limit priority. The default value is 255.
PROJECT	The project number (<9999) to which the console session is to be charged.

NOTE

A special facility to aid password security on half-duplex devices is provided. If a line ends with the keyword PASSWORD or PW the system will type eight splodges at the start of the next line over which the user can type the password. Further data may follow if required.

---

1 Optional.
3 May be repeated.
5 Keyword may not be omitted.

MC

The Phoenix command and JCL catalogued procedure MC has been superseded by PRINT and LIST.

MESSAGE

MESSAGE is a Phoenix command and JCL catalogued procedure which retrieves messages. A line ***MESSAGE FOR YOU is output at a console during the logon sequence if there are any outstanding messages, or at any time during a console session when a message is sent; see the SEND command for details about sending messages.

PHOENIX EXAMPLES

(a) MESSAGE

Any messages for the logged on user are typed and are also available in the %M document.

(b) MESSAGE XYZ9

collects any messages for XYZ9.

(c) MESSAGE TO .B

copies messages for the logged on user to .B.

(d) MESSAGE MNO1 OPT T

tests if there are any messages for user MNO1.

(e) MESSAGE DAY

The message of the day is output.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
(FOR / NAME)	userid or DAY	logged on user or userid in JD	-
TO	dsd	%M	/N
OPT	text	E	-
STORE	(integer / integer K)	36K	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

JCL EXAMPLES

(a) // EXEC MESSAGE

collects messages for the userid under which the job is running.

(b) // EXEC MESSAGE,FOR=ABC1

collects a message for ABC1.

(c) // EXEC MESSAGE,FOR=XYZ1,PRINT1='SPQR.B'

copies a message for XYZ1 to file SPQR.B

(d) // EXEC MESSAGE,OPT=T,FOR=ABC1

tests whether there is a message for ABC1.

(e) // EXEC MESSAGE,FOR=DAY

The message of the day is printed.

JCL PARAMETERS AND DDNAMES

The symbolic parameter FOR defines the userid whose messages are to be collected. Options are passed to the program by the symbolic parameter OPT. Output is normally to the printer, but messages may be filed using the symbolic parameter PRINT1.

OPTIONS

The following options are available:

E Extract; type any messages then delete them from the MESSAGE system. This is the default value when the user identifier following the FOR keyword (explicitly or implicitly) is the same as the logged on user or the userid of the job.

R Read; type any messages, but do not delete them from the MESSAGE system. This is the default value when the user identifier following FOR is different from the logged on user or the userid of the job.

T Test; indicate whether there are any messages for the user.

D Delete; do not type the messages, just remove them from the MESSAGE system.

S Statistics; type the statistics for the message file, including the time and date it was last updated and the user.

RETURN_CODES

A return code of 4 is issued if there are no messages.

A return code of 12 is issued if the MESSAGE file is in use, an unknown option is given, or an attempt is made to delete another user's messages.

A return code of 16 is issued if the MESSAGE file is broken, or there is any other system error.

NOTES

(a) The MESSAGE system does not permit users to delete messages for other users. Messages are printed at reception and then purged from the MESSAGE system after two weeks.

(b) The message of the day is typed during the logon sequence. When a new message of the day is put into the system, the message

    ***NEW MESSAGE OF THE DAY

    is output. The command MESSAGE DAY may be used to type the message.

(c) If the command

    SET TERMINAL NOMESSAGE

    is obeyed, messages other than those from the current session are suppressed.

(d) When a message is sent to several users, the MESSAGE file keeps a copy of the file for each addressee, and when the message is fairly long, this can cause the file to become congested. The recommended procedure, therefore, is to send a one-line message asking the recipient to type a file containing the longer message.

MKINIT

MKINIT is a Phoenix command which allows a user to set up, delete or read a short sequence of commands which will automatically be obeyed whenever he logs on. MKINIT is only available at a console.

PHOENIX EXAMPLES

(a) MKINIT

The current document is set up as the initialisation sequence.

(b) MKINIT SET SPQR SW QUIET

This sets up a single line as the initialisation command.

(c) MKINIT C .CL:INIT

The initialisation sequence contains a call to .CL:INIT

(d) MKINIT TO %O OPT R
C

The initialisation sequence is tested.

(e) MKINIT OPT R TO .TEMPDS:OLDINIT

The initialisation sequence is copied to a member of a partitioned dataset.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Notes
-	text	%C	
USER	text	%USER	
TO	dsd	*	
OPT	text	S or R	S is default unless USER is not logged on userid then R is the default
FROM	dsd	%C	
DATA	text	-	

Note

The available options are SET (S), READ (R) or DELETE (D).

Any key or item may be omitted, unless otherwise stated.

MKINIT
2

NOTES

(a) The text or number of characters in %C must not exceed 59 including new lines. If this number is exceeded a message

USER'S SPACE FULL

is typed.

(b) If DATA is not the last keyword, the text must be in quotes. DATA and FROM are mutually exclusive; if both are given, only the last is used.

MKLIB

MKLIB is a Phoenix procedure which may be used to set up a library of command sequences which may be called with Phoenix command LIBRARY. It may only be used in a job.

EXAMPLE

```
FILE %H! AS .A:MYCOM
SW QUIET SW RESTORE
Any command sequence
!
PARS LIB=.A LM=.LINKLIB MEM=COM2
MKLIB
LIBRARY .LINKLIB:COM2
MYCOM
```

creates a member COM2 of a library .LINKLIB which then allows the members of .A, which are command sequences, to be called using the member names as commands after the LIBRARY command has been obeyed.

PHOENIX PARAMETERS

The symbolic parameter LIB defines the library which contains the command sequences. Output is to a member of a load module library The library is defined by the symbolic parameter LM and the member by MEM For a list of the symbolic parameters type HELP PROC MKLIB at a console

MLI

MLI is a Phoenix procedure and JCL catalogued procedure which invokes the ML/I macro processor ML/I is a general macro processor in the sense that it can be used to process any kind of text. The following documents available from the Computing Service bookshop describe the facilities available and the implementation details:

'The ML/I Users Manual by P.J Brown

'The ML/I Macro Processor - A Simple Introductory Guide' by P.J.Brown

'The Cambridge IBM 370 Implementation of ML/I' by P.Hazel

'MLI Reference Manual' Ed. by P J Brown

PHOENIX EXAMPLES

(a) MLI FROM A

Input is from a catalogued dataset, output is to *

(b) MLI (FROM=%H!) TO=.B:C
Source
!

The source program is from an instream dataset, output is to a partitioned dataset.

PHOENIX PARAMETERS AND DDNAMES

The source is described by the symbolic parameter or ddname FROM. Output is described by the symbolic parameter TO. Options are passed to the program by the parameter OPT Error messages are sent to a dataset identified by the PRINT1 parameter. For a list of the symbolic parameters type HELP PROC MLI at a console.

JCL EXAMPLES

(a)
```
// EXEC MLI
//FROM DD *
 source
//*
//TO DD DSN=&A,DISP=(,PASS),SPACE=(TRK,(3,1)),
// UNIT=DISC
```

Input is from cards, output is to a temporary dataset.

MLI 2

(b)
```
// EXEC MLI
//FROM DD DSN=SPQR.INPUT,DISP=SHR
//TO DD DSN=SPQR.OUTPUT,DISP=(,CATLG),
// SPACE=(TRK,(10,1)),UNIT=DISC
```

Input is from and output is to catalogued datasets.

JCL PARAMETERS AND DDNAMES

The input is taken from the dataset described in a dd statement named FROM, which must be supplied by the user. The output is sent to the dataset described by a dd statement named TO, which must also be supplied by the user. Options may be supplied by using the symbolic parameter OPT. Error messages and informatory messages are sent to the 'SYSPRINT' dataset which is provided in the procedure and set up as SYSOUT=C This can be changed by setting the symbolic parameter PRINT1. For a complete list of the symbolic parameters print the file INFO.JCL PARMS(MLI).

NOTES

(a) The record format may be F, FB, V or VB. Concatenation of unlike datasets is supported

(b) If any of the DCB parameters RECFM, LRECL or BLKSIZE are not specified for the output, they are copied from the input, except in the case of unblocked output, when LRECL is set equal to BLKSIZE or BLKSIZE-4 as appropriate.

OPTIONS

Options should be separated by spaces or commas. The following are available:

SOURCE Causes a listing of the input to be sent to the dataset described by a symbolic parameter named LIST. This is provided in the procedure, and set up as SYSOUT=C (JCL) or * (Phoenix).

PRENV Causes a printout of the current ML/I environment to be output to PRINT1 at the end of processing.

TRAIL Suppresses truncation of trailing spaces from input records.

NO

NO is a Phoenix control command which is synonymous with QUIT 16 i.e. it provides a return code of 16. It is useful in connection with interactive command sequences. See Section 4.10.6 for further details.

EXAMPLE

```
   .
   .
** IS COLOUR BLUE?  ANSWER YES OR NO
C *
IF %RC EQ 16 SKIP NO
* COLOUR IS BLUE
QUIT 100
*NO
   .
   .
   .
QUIT 500
   .
   .
```

If the answer is YES, QUIT 100 is obeyed and control is passed to the calling level of command with a return code of 100; otherwise control is passed to the label NO.

NOTIFY

NOTIFY is a Phoenix command and JCL catalogued procedure which may be used to send a short message to a logged on user. If the user to whom the message is sent is not logged on, the message is lost.

PHOENIX EXAMPLE

(a) NOTIFY SPQR IT'S OPENING TIME!

The message

***FROM ABC1 - IT'S OPENING TIME!

is typed at SPQR's console, if he is logged on.

(b) NOTIFY ABC1 JOB FINISHING

When the above statement is used in a job the message

***FROM J0123.SPQR - JOB FINISHING

is typed at ABC1's console, if he is logged on.

PHOENIX COMMAND SYNTAX

NOTIFY userid text

JCL EXAMPLES

(a) // EXEC NOTIFY,TO=SPQR,TEXT='IT'S OPENING TIME!'

The message

***FROM ABC1 - IT'S OPENING TIME!

is typed at SPQR's console, if he is logged on.

(b) // EXEC NOTIFY,TO=SPQR,TEXT='JOB FINISHING'

When this statement is used, the message

***FROM J0123.SPQR - JOB FINISHING

is typed at SPQR's console, if he is logged on.

JCL PARAMETERS AND DDNAMES

The user identifier to which the message is to be sent is defined by the symbolic parameter TO. The message is defined by the symbolic parameter TEXT.

NOTIFY
2

NOTES

(a) Only the first 60 characters of text are sent, any other characters are lost.

(b) No return codes are given by NOTIFY.

(c) INFO FIND may be used to see if a user is logged on. If he is not logged on SEND should be used instead of NOTIFY.

(d) If the recipient is pressing 'break' when NOTIFY is used, the message is lost.

NULL

NULL is a Phoenix command and JCL catalogued procedure which contains a null program. It can issue a return code. It is useful in JCL for forcing the demounting of a magnetic tape or for pre-allocating datasets.

PHOENIX EXAMPLES

(a) NULL

Any existing return code is unaltered.

(b) NULL 16

A return code of 16 is issued.

JCL EXAMPLES

(a)
```
// EXEC . . .
//ABC DD UNIT=TAPE9,VOL=(,RETAIN,SER=RJS4AZ)
.....
// EXEC NULL
//A DD UNIT=TAPE9,VOL=SER=RJS4AZ
```

NULL is used to force the demounting of a magnetic tape.

(b)
```
// EXEC NULL,RETURN=16
// EXEC . . . COND= . . .
```

NULL is used to issue a return code.

RETURN CODE

The return code to be issued is specified on the NULL command or by the JCL symbolic parameter RETURN, and should be given a value consisting of an unsigned decimal number between zero and 4096. Any other value produces an error message and a return code of zero.

OLDABEND

OLDABEND is a Phoenix flag which may be tested by the user. It is set if any command in the command sequence currently being obeyed (i.e. since the last READY in a console session) abended. It is unset when control returns to the terminal. See Section 4.10.2 for further details.

EXAMPLE

```
        .
        .
        .
    IF OLDABEND ABANDON
    SUBMIT FROM %H!
        .
    !
```

If the OLDABEND flag is set the command sequence is abandoned and control returns to the terminal, otherwise the command sequence continues to be obeyed and a job is submitted.

ONLINE

ONLINE is a Phoenix flag which may be tested by the user. It is set throughout a Phoenix console session.

EXAMPLE

```
IF ONLINE SKIP A
    .
    .
SKIP B
*A
    .
    .
*B
```

If the command is being obeyed during a Phoenix console session, control is transferred to label A, otherwise the command stream continues to be obeyed.

PARS

PARS is a Phoenix command which sets or unsets symbolic parameters globally. The setting takes effect immediately. PARS is synonymous with GLOBAL, SET P and SET PARS.

PHOENIX EXAMPLES

(a) PARS U=SPQR PRINTER 2K -LIMSTORE ROUTE CENTRAL P=123

This sets parameters U, PRINTER, ROUTE and P and unsets LIMSTORE.

(b) SET PARS X 'text' Y=12 Z='' -A
C .P.COMMANDS:C1

Symbolic parameters are set for a command sequence.
X is set to text, Y to 12, Z to the null string; A is unset.

(c) PARS A=ABCD
**<A>

If these two commands are in a command sequence with no local value of A set then the global setting of A as ABCD will cause substitution of ABCD in the following comment line.

(d) PARS R='SPQR',2

This sets the parameter R to SP.

(e) GLOBAL %G1=%L4-%QLEVEL

%G1 is set to the difference of %L4 and %QLEVEL.

PHOENIX COMMAND SYNTAX

```
GLOBAL
SET P
SET PARS   parameter-list
PARS
```

where parameter-list is any number of items of the form

```
( parameter-name = value          )
( - parameter-name                )
  parameter-name =  value,length
```

where = is optional
See Section 4.9 for further details.

PARS special-parameter assignment

See Variables for further details.

Choose one of the alternatives in brackets.

PARS
2

NOTES

(a) Parameters set by PARS apply to all levels of command and take effect at the end of the command.

(b) Up to 36 alphanumeric single character parameters are allowed. Multicharacter parameters may be up to eight characters long; there may not be more than 64 of these.

(c) Value may be any sequence of characters; if spaces are included the value must be enclosed in quotes, and any quotes in the string must be represented by two quotes.

(d) The total length of all values set at any time may not exceed 498 characters.

(e) All parameters, except the preset ones (U and S), are unset by use of the CLEAR command. Individual parameters are unset by preceding the parameter name with a - in the PARS command.

PASSWORD

PASSWORD is a Phoenix command and JCL catalogued procedure which provides facilities for checking and altering users' passwords and for interrogating and altering values of the password controls for users and projects. There are four types of access which may require the quoting of a correct password; these are:

LOGON access
JOB access
JOBQUEUE access
PASSWORD access

A password consists of up to 8 alphanumeric characters.

PHOENIX EXAMPLES

(a) PASSWORD CHECK SECRET

This checks the password of the current user and, if the password is correct, puts the job or console session into 'password checked' state. If the password is incorrect the state of the job or session is unchanged.

(b) PASSWORD NEW HUSHHUSH

This changes the value of the current user's password. Passwords may only be changed when the job or console session is in 'password checked' state.

(c) PASSWORD CHECK NOSECRET NEW NOONENOS ALL CHECKED

This quotes the password of the current user, then changes the password to a new value 'NOONENOS'. Finally the value of all the password controls is set to be checked.

PHOENIX COMMAND SYNTAX

PASSWORD TO dsd[5] keyword-value-pair[3]

---

3 May be repeated.
5 Keyword may not be omitted.

PASCAL

PASCAL is no longer supported.

PASSWORD
2

OPTIONS

In the notes below the current user and current project are the userid and project number under which the job or console session is running.

Keyword	Values	Notes
ALL	(CHECK / UNCHECKED / FORBID)	Sets value of all password controls for the current user.
CHECK	password	Checks the password of the current user. If the password is correct the job or console session is put into 'password checked' state. If the password is incorrect the state of the job or session is unaltered.
(JOB / JOBS)	(CHECK / UNCHECKED)	Sets the value of the JOB password control for the current user.
(JOBQUEUE / JOBQ)	(CHECK / UNCHECKED / FORBID)	Sets the value of the JOBQUEUE password control for the current user.
LOGON	(CHECK / UNCHECKED)	Sets the value of the LOGON password control for the current user.
NEW	password	Changes the value of the current user's password. The job or console session must be in 'password checked' state to use this.
PROJECT	number	All subsequent keyword-value pairs apply to a given project. (The default value is the current project). Only the JOB and LOGON fields apply to projects. The user must have access to the project to use this facility.

Choose one of the alternatives in brackets.

NOTES

(a) If a keyword which takes a password as its value appears at the end of a line, and the command is being run at a console, then eight splodges are written to the console and input may be continued on the next line.

(b) If no parameters are given, or if no further data follows a PROJECT keyword, the values of the password controls for the relevant user or project are output.

(c) A job or console session must be in 'password checked' state when the value of a password or password control are changed. The values of the password controls are output after such a change.

(d) For LOGON and JOB access, the password controls of both the user and the project are examined, and the more restrictive value taken. For JOBQUEUE access, only the password control of the job's userid is used.

RETURN CODES

Any error causes a suitable message to be output, and a return code of 8 to be issued.

PATCH

PATCH is a Phoenix command and JCL catalogued procedure which allows disc datasets, including partitioned datasets containing load modules, to be interrogated and updated. The dataset may have any record format. A full specification of PATCH is given in 'Cambridge Aids to Assembler Programmers'.

PHOENIX EXAMPLES

(a) PATCH .FILE
patch commands
Q

This patches .FILE.

(b) PATCH .LINK WITH .PATCHES

Commands are taken from the file .PATCHES.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
FILE	dsd	-	-	input
WITH	dsd	*$	/R	patches
TO	dsd	*	/N	verification
DD	dsdlist	%N	/N	-
OPT	text	-	-	-
STORE	(integer / integer K)	40K	-	-

JCL EXAMPLES

(a)
```
// EXEC PATCH,FILE='SPQR.FILE'
//SYSIN DD *
 patch commands
//*
```

This patches SPQR.FILE.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the given order.

PATCH
2

(b) // EXEC PATCH,FILE='SPQR.OLD',
// WITH='SPQR.PATCHES'

Patch commands are taken from the file .PATCHES.

JCL PARAMETERS AND DDNAMES

The file to be patched is identified by the symbolic parameter FILE or by ddname FILE.

Patch commands are taken either from the dataset defined by the symbolic parameter WITH or from the dataset defined by ddname SYSIN.

Verification is normally sent to the printer, but may be sent to a dataset using the symbolic parameter TO.

For a full list of the symbolic parameters for PATCH print the file INFO.JCL.PARMS(PATCH).

NOTE

For further details of PATCH see 'Cambridge Aids to Assembler Programmers'.

PDSRECOV

PDSRECOV is a Phoenix command and JCL catalogued procedure which copies a partitioned dataset, also copying the 'dead space', assigning names to these 'dead members' so that previously deleted or overwritten members may be accessed. PDSRECOV may not be used on a load module pds.

PHOENIX EXAMPLES

(a) PDSRECOV .A TO &A

All members of .A, including deleted and overwritten members are copied to &A.

(b) PDSRECOV PDS .B TO .C NAME ABC

All members of .B are copied to .C; the recovered members being given the names ABC001, ABC002 etc.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
PDS	dsd	-	/R	mandatory item
TO[5]	dsd	-	/N	mandatory item
VER	dsd	*	/N	-
NAME	text	Z#	-	member-name prefix
STORE	(integer / integer K)	24K	-	-

JCL EXAMPLES

(a) // EXEC PDSRECOV,PDS='SPQR.A',TO=&A

All members of SPQR.A, including deleted and overwritten members are copied to &A.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

PDSRECOV
2

(b) // EXEC PDSRECOV,PDS='SPQR.B',TO='SPQR.C',NAME=ABC

All members of SPQR.B are copied to SPQR.C, the recovered members being given the names ABC001, ABC002 etc.

JCL PARAMETERS AND DDNAMES

The input pds is defined by the symbolic parameter PDS or by a dd statement with ddname FROM. A destination pds should be defined by the symbolic parameter or ddname TO. A name for deleted members may be passed to the program using the symbolic parameter NAME. Verification output and messages are defined by the ddname SYSPRINT.

NOTES

(a) If text is supplied to the program by the NAME symbolic parameter, this character string is used to start the madeup names. The names are padded to a maximum of eight characters, by means of a serial number of up to three digits. e.g. If the NAME text is ZZZ, the madeup names will be ZZZ001, ZZZ002 etc. The default character string is Z#.

(b) PDSRECOV will not work for a load module pds, or more specifically, for a pds with TTRN fields in the user data field.

(c) The program deals with aliases by supplying a madeup main name for any floating alias names. It will also copy user data in the directory entry. If a malformed TTR (pointer) is found in the directory, a message is output, together with the corresponding member name, and that directory entry is ignored.

RETURN CODES

The program issues the following return codes:

0	Successful completion.
4	The directory is malformed; an entry was found pointing to somewhere other than the start of a member.
8	The input dataset is not a pds. Name clash in output directory.
12	TTRNs in user data (i.e. input pds is a load module pds). User data in directory entry too long. Insufficient directory blocks in TO.
16	The input or output pds's cannot be opened.

PDSSCAN

PDSSCAN is a Phoenix command and JCL catalogued procedure which reads a pds, printing out the TTR (pointer) and the first line or so of each member, irrespective of whether that member appears in the directory. PDSSCAN is useful when investigating a broken pds.

PHOENIX EXAMPLES

(a) PDSSCAN .PDS

The dataset .PDS is examined, and the TTR and first line of each member printed.

(b) PDSSCAN FROM .BROKEPDS VER &A

The pds .BROKEPDS is examined and verification of the TTR and the first line of each member filed to &A.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
FROM	dsd	%N	/R
VER	dsd	*	/N
STORE	(integer / integer K)	16	-

JCL EXAMPLES

(a) // EXEC PDSSCAN,FROM='SPQR.PDS'

The TTR and first line of each member is printed.

JCL PARAMETERS AND DDNAMES

The source dataset is defined by the symbolic parameter FROM. Verification and messages are sent to the dataset defined by the ddname SYSPRINT.

PL/I

PL/I is a high level general purpose language that may be used with the three compilers, F, Checkout and Optimising which are accessed using either Phoenix procedures or JCL catalogued procedures.

COMPILERS

F	An old IBM compiler for a slightly restricted subset of the PL/I language which should not normally be used except for existing PL/I F programs.
Checkout	IBM Program Product debugging compiler. It has good compilation and execution diagnostics, compiles 10 times faster than the optimiser, but executes up to 100 times more slowly. It should always be used for debugging.
Optimising	IBM Program Product. Very slow in compilation, but produces fast code.

Both the Checkout and Optimising compilers accept the language defined in the IBM publication 'PL/I Checkout and Optimising Compilers: Language Reference Manual' SC33-0009.

As routines compiled by the two compilers may be freely mixed, routines under development should be compiled with the Checkout compiler, and only working routines compiled with the Optimiser. Further details of the IBM compilers may be found in the publications:-

'PL/I Checkout Compiler : Programmer's Guide'	SC33-0007
'PL/I Checkout Compiler : Messages'	SC33-0034
'PL/I Optimising Compiler : Programmer's Guide'	SC33-0006
'PL/I Optimising Compiler : Messages'	SC33-0027

The Checkout and Optimising compilers and associated libraries are designated proprietary, and users are referred to the Introduction to Part II of this manual.

PROCEDURES

The following JCL and Phoenix procedures are available:-

For the F compiler:	PL1LFC, PL1LFCL, PL1LFCLG, PL1LFLG
For the Optimising compiler:	PLIXC, PLIXCL, PLIXCLG, PLIXL, PLIXLG, PLIXG
For the Checkout compiler:	PLICKC, PLICKCL, PLICKCLG, PLICKLG, PLICKR (one-step compile and go), PLICKI (load compiled program and go)

Note that the Phoenix procedures are not available for use in a Phoenix console session.

PL/I
2

PHOENIX EXAMPLES

(a) PLICKR PROGRAM=.PROG

This is load-and-go running of the program contained in the catalogued dataset .PROG.

```
(b)  PLICKR (PROGRAM=%H#)  LISTC=S
      P: PROC
      . . . .
      END P;
     #
```

The program is given in a here dataset and a source listing is requested using the symbolic parameter LISTC.

(c) PLICKCL PROGRAM=.PROG NAMEL=.LIB:P TEXT=.TEXT

A procedure is compiled by the checkout compiler and link edited with the load module output being sent to the library .LIB and the intermediate text being stored in the partitioned dataset .TEXT.

```
(d)  PLICKLG (LKIN=%H#  DATA=.IN  LIB=.LIB) TEXT=.TEXT
      INCLUDE LIB(P)
     #
```

The previously compiled procedure is run supplying the intermediate text with symbolic parameter TEXT and including the load module stub by using a link edit control statement. Data for the program is provided within the environment with a data definition with ddname DATA.

(e) PLIXCLG PROGRAM= PROG

A program defined by the symbolic parameter PROGRAM is compiled by the Optimising compiler, link edited and run. Note that the Optimising compiler does not produce any intermediate text.

Further examples may be found in Example sheets 5, 6 and 7.

PHOENIX PARAMETERS AND DDNAMES

Complete lists of Phoenix parameters for each procedure may be obtained by using the command 'HELP PROC procedure-name'.

In all the compilation procedures, the program must be given by either the symbolic parameter or the ddname PROGRAM; and program listing options can be altered by the use of the symbolic parameter LISTC(default value NS).

PL/I
3

The symbolic parameter TEXT may be used with procedures PLICKC, PLICKCL, PLICKCLG and PLICKLG to define the intermediate text partitioned dataset.

Extra load module libraries to be used in the automatic library scan by the linkage editor are given using the LIBRARY symbolic parameter.

Program parameter information is given using the symbolic parameter PARMG while ISAG may be used with PLIXCLG, PLIXLG and PLIXG for optimisation of further runs (for more details see Example Sheet No. 5).

JCL EXAMPLES

(a)
```
// EXEC PLICKR
//PLI.SYSIN DD DSN=SPQR.PROG,DISP=SHR
```

This is load-and-go execution of the program in SPQR.PROG by the checkout compiler.

(b)
```
// EXEC PLICKR,LISTC=S
//PLI.SYSIN DD *
   P:PROC
      END P;
//*
```

The program is given on an instream dataset and a source listing is requested using the LISTC symbolic parameter.

(c)
```
// EXEC PLIXCLG
//PLI.SYSIN DD DSN=SPQR.PROG,DISP=SHR
```

The program is run using the Optimising compiler.

The stepname for the PL1F compiler is PL1L.
Further examples may be found in Example Sheets 5, 6 and 7.

JCL PARAMETERS AND DDNAMES

Complete lists of JCL symbolic parameters may be obtained from the partitioned dataset INFO.JCL.PARMS with the member name corresponding to the procedure name. In all the compilation procedures the program must be given on a dd statement with ddname SYSIN, and program listing options can be altered by the use of the symbolic parameter LISTC (default value NS). The dd statement SYSITEXT may be used with procedures PLICKC, PLICKCL, PLICKCLG and PLICKLG to define the intermediate text partitioned dataset. Extra load module libraries may be used in the automatic library scan by the linkage editor by overriding the SYSLIB dd statement.
Program parameter information is given using the symbolic parameter PARMG while ISAG may be used with PLIXCLG, PLIXLG and PLIXG for optimisation of further runs (for more details see Example Sheet No. 5).

PL/I
4

LIBRARIES

Library	PDS	Built into procedures	Notes
PLIX resident library	PPLIB.PLIBASE	Yes	Contains all resident system routines including those for mixing with PLICK.
PLICK transient library	PPLIB.PLITRLIB	Yes	Seldom used routines which are loaded dynamically when required.
Checkout compiler transient library	PPLIB PLICLIB	Yes	Holds interpreter routines which are loaded dynamically.
FORTRAN linking library	S.FORTLIB	No	

NOTES

Advantage should be taken of the store management report in order to optimise further runs. See Example Sheet 5.

Fortran routines may be mixed with PL/I programs although great care must be taken with variable types in the arguments.

The routine must be declared suitably as an External Fortran routine and the appropriate library scanned in the link edit step. The Fortran system library SYS1.FORTLIB will often be needed as well, in particular if the routine to be called does any input/output or uses Fortran supplied mathematical routines. A smaller version of SYS1.FORTLIB may be used, S.FORTLIB, which contains the mathematical routines and special versions of #IBCOM and #ERRMON (the Fortran input/output package and error monitor respectively). These special routines produce an ABEND and a suitable message if called. This means that a Fortran error will give a message on the HASP log and/or a PL/I message (which will state the type of error and the fact that it occurred in a non PL/I routine).

The following comments apply only to the NAG Library.

If the soft fail option in the NAG library (see the NAG documentation for details) is used, S.FORTLIB must also be scanned. If a Fortran error occurs (which should be rare), it will usually have been caused by the passing of incorrect parameters to the NAG routine and this should

be checked first. If it is necessary to obtain the full Fortran error message, SYS1.FORTLIB should be scanned instead of S.FORTLIB, the region size for the GO step increased by at least 24K and a DD statement for GO.FT06F001 output provided. If the hard fail option is used, the procedure described above to obtain error messages should be employed.

Examples: the examples shown call the NAG routine S07AAF.

(a)
```
// EXEC PLICKCLG
//PLI.SYSIN  DD *
.
  DCL  S07AAF EXTERNAL ENTRY (DEC(16),FIXED BIN(31))
   OPTIONS(FORTRAN)   RETURNS(DEC(16));
.
   T = S07AAF(X,IFAIL);
.
//*
//LKED.SYSLIB DD
//            DD  DSN=SYS2.NAGLIB,DISP=SHR
//            DD  DSN=S.FORTLIB,DISP=SHR
```

uses the soft fail option.

(b)
```
PLICKCLG (PROGRAM=%H#  FT06F001=*C
) LIBRARY=SYS2.NAGLIB+SYS1.FORTLIB
program as above
#
```

uses the hard fail option under Phoenix.

ERRORS

A comprehensive discussion of error handling in PL/I may be found in the manual 'Error Tracing and Debugging Aids'.

POPS

The Phoenix command and JCL catalogued procedure POPS has been superseded by the utility FILEPDS.

POPS
2

PRINT

PRINT is a program for converting sequential input to a multi-column format. Although the output will normally be sent to a printer, the program can be used to restructure datasets. Input may come from a sequential dataset (including a single member of a partitioned dataset), or from a partitioned dataset. In the latter case options are provided to include or exclude particular members.

The normal action of PRINT is to fill the output page as densely as possible. Various options are provided which enable the user to control the layout of the output.

PHOENIX EXAMPLES

(a) PRINT .A

The dataset .A is output to the printer, if the command is issued in a job, and to the console if it is issued during a console session.

(b) RUN PRINT .PROGS:SUBA OPT MC=3

Member SUBA of the dataset .PROGS is printed on the line printer. A maximum of three columns is printed on any page.

(c) " RUN PRINT .PROG OPT S=6 " STATUS "

The dataset .PROG is output to the line printer in multicolumn format. The minimum separation between columns is 6 spaces. A STATUS command is issued to check the progress of the job.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
FROM	dsd	-	/R
TO[5]	dsd	*	/N
WITH[5]	dsd	-	-
VER[5]	dsd	*C	-
STORE	(integer / integer K)	-	-
OPT	text	-	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Key may not be omitted.

JCL EXAMPLES

(a) // EXEC PRINT,NAME='SPQR.A'

The dataset SPQR.A is printed on the line printer in multicolumn format.

(b) // EXEC PRINT,NAME='SPQR.B(C)',OPT='MC 3'

Member C of pds SPQR.B is printed on the line printer. The maximum number of columns on a page is 3.

JCL PARAMETERS AND DDNAMES

The dataset to be printed is defined by the symbolic parameter NAME or ddname FROM. The output dataset is defined by the symbolic parameter TO or ddname TO and errors are sent to the dataset defined by the ddname SYSPRINT. Options may be passed to the program via, firstly the symbolic parameter OPT, and secondly by the dataset defined by ddname SYSIN.

OPTIONS

Options consist of keywords, sometimes followed by arguments. They may be separated by space, comma and/or end of line. Only the first eight characters of keywords are significant. The character '=' may optionally appear between a keyword and its argument. Keywords may be given in upper, lower or mixed case. The types of argument are:-

number - a sequence of digits
name - a sequence of up to eight alphanumeric characters or ?, &, !, $, # or @, beginning with a non-digit
string - any characters, enclosed in matching single or double quotes
number list - a list of numbers separated by space, comma or end of line and enclosed in brackets
name list - a list of names separated by space, comma or end of line and enclosed in brackets

Options are processed in the order in which they are read, and the later one used if two conflict. There are two sets of options; those which apply only to partitioned datasets, and those which always apply.

PRINT
3

General options

COL=number (abbreviation C)

Specifies that output is to be in C columns of fixed, equal width MCW=(W-(C-1)*S)/C and that there is to be no spreading (this only affects the last page).

CC/NOCC

NOCC specifies that control characters in the input are to be ignored, that is, treated as "advance one line". The default for this option is CC if the input dataset contains control characters, otherwise NOCC. (See also OVERPRINT, SKIPOVERPRINT.)

COLFACTOR=number (abbreviation CF)

Specifies the maximum column width MCW as a fraction of the linewidth W. The exact formula is MCW=(W-(CF-1)*S)/CF. This ensures that at least CF columns are output on each page.

COLTITLE[=string]

COLTITLE requests a heading at the top of each column and optionally specifies the string to be used as a title. However, if any column on a page is too narrow, a single heading is output at the top of the page. If the PAGENUMBER option is on, a different number is output for each column. See also TITLE.

COLWIDTH=number (abbreviation CW)

Specifies fixed columns of width CW characters. CW may not exceed W.

FLOWINDENT=number (abbreviation FI)

Specifies the number of spaces to be inserted at the start of the overflowing parts of lines which are longer than MCW. It must be less than MCW, and defaults to zero.

LINENUMBERS/NOLINENUMBERS (abbreviations LN/NOLN)

LINENUMBERS requests that a line number be output with each line. The default is NOLINENUMBERS. The line number and following spaces occupy 7 character positions, and this must be included when setting the maximum column width. The line number is omitted from overprinting lines, and from continuation lines. When a partitioned dataset is used as input, the line number is reset at the start of each member.

LINEOFFSET=number (abbreviation LO)

Specifies the number of spaces by which all lines are to be offset from the left-hand side of the physical output page (default zero). This count is not included in W. The sum of W and LO must not exceed the effective LRECL of TO (or the terminal width).

PRINT
4

LINEWIDTH=number (abbreviations W, LW)

Specifies the maximum width of line to be output. This must not be greater than the effective LRECL of the output dataset less the line offset LO, to which value it defaults. A value of zero requests the default. If output is to Phoenix terminal, the terminal width less the line offset is used as the default (and maximum).

MAXCOL=number (abbreviation MC)

Specifies the maximum number of columns to be printed on any page. This must be less than 255. The default value is 1 if PAGELENGTH (L) is 1; otherwise it is 12. A value of zero requests the default.

MAXCOLWIDTH=number (abbreviation MCW)

Specifies the maximum column width MCW in characters. This must not be greater than W, to which it defaults.

NOCC - see CC

NOLINENUMBER - see LINENUMBER

NOOVERPRINT - see OVERPRINT

NOSPREAD - see SPREAD

NOTITLE - see TITLE

NOTRUNCATE - see TRUNCATE

NOUNDERLINE - see UNDERLINE

OVERPRINT/NOOVERPRINT (abbreviations OP/NOOP)

NOOVERPRINT causes all overprinting control characters to be converted to "advance one line". The default is OVERPRINT. (See also CC, SKIPOVERPRINT.)

PAGECC=string (abbreviation PCC)

Specifies the ANSI control character to be used at the start of each output page. The string must contain exactly one character. The default is " " if the value of L (the number of lines per page) is 1, otherwise it is "-" when output is to Phoenix terminal and "1" for other devices. By using both PAGECC and PAGELENGTH it is possible to print a number of logical pages on one physical page.

PAGELENGTH=number (abbreviation L)

Specifies L, the number of lines per output page. A value of zero requests the default. If the TITLE option is on, the heading line and following blank line are included in L. If the program is run offline, the default value is the Hasp PAGE parameter or 60 if this is zero. If the program is run online the default is 1 if output is to terminal, otherwise 60.

PRINT
5

PAGENUMBER[=number] (abbreviations PN/NOPN)

Specifies that page numbers are to be included in the page headings, and optionally gives the first page number (default 1). A value of zero is permitted. One of the TITLE or COLTITLE options must be set for PAGENUMBER to take effect. The default is NOPAGENUMBER but note that INDEX sets PAGENUMBER.

PAGEWAIT[=string]/NOPAGEWAIT (abbreviation PW/NOPW)

If output is to a Phoenix terminal, PAGEWAIT causes PRINT to wait for the user to type a carriage return before starting the second and subsequent pages. The message "page waiting" is output if a string is not supplied.
If a string is supplied, it is used instead. The default for this option is NOPAGEWAIT. (See also VDU.)

SEPARATION=number (abbreviation S)

Specifies S, the separation between columns as a number of spaces. A value of zero is permitted. The default value is 2. Unless the NOSPREAD option is on, the value of S is used as a minimum separation and actual separations may be greater (see 3.4 for details).

SAMPLE=number list (abbreviation SA)

The numbers must be in ascending order. They are taken in pairs, and specify by line number inclusive ranges of lines to be output. The character '*' may be used instead of a number to refer to the end of the input. A single number at the end of the list causes a single line to be output. If the input is from a partitioned dataset, the SAMPLE option applies to each member independently.

SKIPOVERPRINT (abbreviation SKOP)

This option causes overprinting lines to be omitted from the output. (See also CC, OVERPRINT.)

SPREAD/NOSPREAD

NOSPREAD requests that all the columns on a page be separated by exactly S spaces. This may leave substantial space on the right of the page. SPREAD requests that S be treated as a mimimum separation (see 3.4 for details). The default is SPREAD except when output is to a Phoenix terminal, when it is NOSPREAD. Note that the COL option sets NOSPREAD.

TITLE[=string]/NOTITLE

TITLE requests a single heading at the top of each page, and optionally specifies the string to be used as a title. The date and time are included in the heading together with the page number if the PAGENUMBER option is on. The default title string is the name of the dataset, including the current member name. Temporary dataset names are given with a single '&'. The default for this option is TITLE, except when output is to Phoenix terminal. See also COLTITLE.

PRINT
6

TRUNCATE/NOTRUNCATE (abbreviations TR/NOTR)

TRUNCATE specifies that lines longer than the maximum column width CW are to be truncated; NOTRUNCATE (the default) specifies that they are to be split into several lines.

UNDERLINE/NOUNDERLINE (abbreviations U/NOU)

UNDERLINE specifies underlining for page headings, column headings, and partitioned dataset member headings. The default is NOUNDERLINE.

VDU

This is just a short way of specifying PAGELENGTH=22, NOOVERPRINT and PAGEWAIT, which are suitable options when output is to a vdu terminal.

Pds options

The following options are only applicable when the input is a whole partitioned dataset:-

ALIASES/NOALIASES (abbreviations A/NA)

ALIASES (the default) requests that alias names be printed at the head of each member, following the main name. NOALIASES is forced if NONAMES is set.

COLMEMBER (abbreviation CM)

This option causes each member to start at the head of a new column. It is precisely equivalent to MEMBERCC='1', described below. (See also PAGEMEMBER.)

EXCLUDE=name list (abbreviation E)

The named members are omitted from the output. Reference to either an alias or a main name is sufficient to exclude a member. If EXCLUDE is given, INCLUDE may not be given. Failure to find an excluded member is not considered to be an error.

FIRSTMEMBER=name (abbreviation FM)

This option is only permitted if members are being output in alphabetic order. After any other selection options have been applied (INCLUDE, EXCLUDE) any members whose names precede the given name in collating sequence are omitted.

INCLUDE=name list (abbreviation I)

Only the named members are included in the output. Reference to either an alias or a main name causes the member to be included. If INCLUDE is given, EXCLUDE may not be given. Failure to find an included member is considered to be an error. Included members are ouptut in alphabetic order. (See also ORDER.)

PRINT
7

INDEX/NOINDEX

INDEX requests that an index giving member names and page numbers be output at the end. It also requests page numbers if headings are being output (see PAGENUMBER). The default is NOINDEX. If ALIASES is set alias names are included in the index in their alphabetic position, with a reference to their main name.

LASTMEMBER=name (abbreviation LM)

This option is only allowed if members are being output in alphabetic order. After any other selection options have been applied (INCLUDE, EXCLUDE) any members whose names follow the given name in collating sequence are omitted.

MEMBERCC=string (abbreviation MCC)

Specifies the ANSI control character to be used for the first line of each member. The string must contain exactly one character. If the NAMES option is on, the control character is used for the first heading line of each member, otherwise it is used for the first data line. The default value is "-" if the NAMES option is on, otherwise the actual control character read from the data line is used. When member headings are being output the control character of the first data line is forced to "0", ie one blank line is left after the heading. Setting MEMBERCC="1" (or COLMEMBER, which is equivalent) causes each member to begin at the head of a column, whether the NAMES option is on or off. If the NAMES option is on, a new column is started if there are fewer than 7 lines left in the current column, whatever the value of MEMBERCC.

NAMES/NONAMES (abbreviations N/NN)

NAMES (the default) requests that member names be output at the start of each member. NONAMES forces NOALIASES.

NOALIAS - see ALIAS

NOINDEX - see INDEX

NONAMES - see NAMES

ORDER=name list (abbreviation O)

This behaves as INCLUDE, with the additional effect that the included members are output in the order in which they appear in the name list.

PAGEMEMBER (abbreviation PM)

This option causes each member to start at the beginning of a new page. (See also COLMEMBER.)

PRINT
8

(a) The standard action of PRINT is to read input lines and format them into columns of length L. Lines that are longer than MCW are split into more than one line; if such lines are overprinted, the overprinting lines are also suitably split. When the number of columns exceeds MC or the sum of the column widths and separators exceeds W a suitable number of columns are output. These are spread evenly across the page if possible, S being treated as a minimum separation.

(b) By default, unless the output is to a Phoenix terminal, a heading is output at the top of each page. The heading consists of the dataset name (including the current member name if the input is from a partitioned dataset) and the current date and time. It is output left justified. The TITLE, COLTITLE and PAGENUMBER options can be used to change the default action. The PAGECC option can be used to specify the control character used at the start of each page.

(c) Because of the possibility of encountering following overprinting lines, a line normally has to be read into store in its entirety before being processed. This is an embarrassment when processing files with very long lines, as occur for example in VBS files. To overcome this problem PRINT reads part lines when it knows that there is no possibility of following overprinting records. This is the case if the NOCC option is set, which is the default if the input file does not contain control characters.

(d) All output takes place on a virtual page of width W, which does not include the line offset LO. A non-zero line offset may be used to centre output on the physical page or to create a left-hand margin (eg for binding purposes). Right-hand margins can be obtained by suitable settings of the MAXCOL and COLWIDTH or MAXCOLWIDTH options.

When a page is to be output, the columns are normally spread across it as evenly as possible. This happens after the decision has been made to output a given number of columns on a page, hence it is known that the columns will fit with separation S. The spreading algorithm is:-

(i) Let n be the number of columns to be output, w[i] be the width of the ith column, mx[i], mn[i] be the maximum and minimum widths of the first i columns respectively.

(ii) If 2*mn[n] <= mx[n] then goto (iv). This prevents lopsided effects when there is a great disparity in column widths.

(iii) If n*(mx[n-1]+S) <= W and W/n >= mx[n] then all columns are output in equal fields of width W/n.

(iv) Otherwise each column is output in a field of width w[i]+max(S,(W-w[1]-w[2]-...w[n])/n).

The NOSPREAD option can be used to stop column spreading. If it is set each column is output in a field of width w[i]+S. NOSPREAD is the default if output is to a Phoenix terminal.

PRINT
9

(e) By default, columns of varying width are read, the width of a column being the width of its widest line (which cannot exceed MCW). The COL and COLWIDTH options can be used to request columns of fixed width. Short lines are then notionally padded out with spaces to width MCW.

(f) PRINT usually produces the whole of its output without stopping. There is however the PAGEWAIT option, applicable only when output is to a Phoenix terminal, which causes the program to wait for user interaction after each page. As a convenience, the option VDU is also provided. This sets PAGEWAIT and NOOVERPRINT, and also sets the pagelength to 22.

(g) The four options for specifying the maximum column width with or without fixed width columns (COL, COLFACTOR, COLWIDTH and MAXCOLWIDTH) are mutually exclusive. A value of zero for any of them requests the default maximum column width, ie W.

(h) When output is to a Phoenix terminal, BREAK followed by carriage return causes all previously read lines to be abandoned. If the input is a partitioned dataset, the rest of the current member is also abandoned.

RETURN_CODES

The following return codes are issued by PRINT:-

```
 0 - successful execution.
 8 - all errors except those below.
12 - error in reading pds directory.
   - INCLUDE or EXCLUDE specified for sequential file.
   - failure to find included member.
16 - insufficient store.
   - error on TO dataset.
   - FROM missing.
   - open failure on FROM or SYSIN.
   - program and system errors.
```

PROJECT

PROJECT is a Phoenix command which may be used to set and unset a project as a default project or to give information about a particular project. The values printed are:

S the number of shares
U the usage of the project
V the work done per share
W the amount of work in the machine.

If no project number is given, the project number associated with the console session or job is used.

PHOENIX EXAMPLES

(a) PROJECT

The values of S, U, V, W for the project associated with the console session or that under which the job is being run are output.

(b) PROJECT 9999 9997

The values of S, U, V, W for projects 9999 and 9997 are printed.

(c) PROJECT 9234 DEFAULT

The default project for the logged on userid or userid under which the current job is running, is set to 9234.

(d) PROJECT DEFAULT

The default project is set to the current project, i.e. the project number under which the console session or job is running.

(e) PROJECT NODEFAULT

The default value of PROJECT is cancelled.

(f) PROJECT TYPEDEFAULT

The value of the default project is typed.

PROJECT
2

PHOENIX COMMAND SYNTAX

To examine a project's scheduling values

PROJECT [TO dsd] n1[13]

where n1 is a project number.

To set a default project

PROJECT n1[1] DEFAULT

To unset a default project

PROJECT NODEFAULT

To type the default project

PROJECT TYPEDEFAULT

---

1 Optional.
3 May be repeated.

PROTECT

PROTECT is a Phoenix command which may be used to protect individual permanent files from unauthorised or accidental deletion, overwriting etc. The protection status of new files is specified by the CREATION command or if this has never been used for the filespace it defaults to FF. The protection status of an individual file may also be altered by use of the EXAMINE command.

PHOENIX EXAMPLES

(a) PROTECT SPQR.PROGS

The dataset SPQR.PROGS is given the protection status 'free' for the filespace manager and the filespace relations, and 'read only' for all other users. .

(b) PROTECT FILE SPQR.LIB STATUS UR

The file SPQR.LIB is given the protection status 'update' for the filespace manager and the filespace relations, and 'read only' for all other users.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
FILE	dsd	-	-	Mandatory item
STATUS	text	FR	-	-
VER[5]	dsd	*	/N	

NOTES

(a) There are three protection levels

F - Free access; the dataset may be altered, read or deleted
U - Update; the dataset may be altered, but not deleted
This is useful for pds's
R - Read only

Any key or item may be omitted, unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

PROTECT
2

(b) There are two groups of users recognised by the protection system:

1. The filespace manager (normally the fileowner) and the filespace relatives, as created by the RELATION command.

2. All other users.

The two letters of the protection status refer to these two groups respectively.

PUNCH

PUNCH is a Phoenix command and JCL catalogued procedure which may be used in jobs to punch a sequential dataset (which may be a member of a pds) on the card punch. At a console PUNCH is equivalent to TYPE.

PHOENIX EXAMPLES

(a) PUNCH .A

punches a catalogued dataset.

(b) PUNCH %H! OPT=S(RJS,10,10)
data
!

punches instream dataset with sequence numbers.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
FROM	dsd	%C	/R
OPT	text	-	see OPTIONS below
STORE	(integer / integer K)	32K	-

JCL EXAMPLES

(a) JOB RJS5 224,PUNCH 50
// EXEC PUNCH,NAME='RJS1.PROGX'

punches a catalogued dataset.

Choose one of the items in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

(b)
```
JOB RJS6 201,PUNCH 500
//  EXEC PUNCH,OPT='S(RJS,10,10)'
//FROM DD DATA
    data
/*
```

repunches an instream dataset with sequence numbers.

JCL PARAMETERS AND DDNAMES

If the source is a catalogued dataset, the source is defined by the symbolic parameter NAME in the calling statement. If not, the source must be defined by an overriding dd statement named FROM.

OPTIONS

The parameter OPT may be used to specify other options as in FILE.

NOTE

The Phoenix command PUNCH cannot be used to punch paper tape; FILE should be used instead.

QUIT

QUIT is a Phoenix control command which returns control to the calling level of commands. If the command is not nested, control returns to the console or the job is terminated.

The command may be followed by an integer which sets a return code.

See Section 4.10.6 for further details.

EXAMPLE

```
        .
        .
        .
    IF ERROR QUIT 200
        .
        .
        .
```

If the ERROR switch is set control is passed to the calling level of commands.

RC

RC is a Phoenix flag which is set by the system and may be tested by the user It is set if the last command returned a non-zero return code and is unset by the next non-control command that terminates with a zero return code. CC is synonymous with RC. See Section 4.10.2 for further details.

EXAMPLE

```
        .
        .
    UNLESS RC SKIP ON
        .
        .
        .
    *ON
```

If a return code greater than zero was issued by the previous command the jump is not taken and processing continues; otherwise control is transferred to the label ON.

REQUEST

REQUEST is a Phoenix control command which is used to demand values of symbolic or special parameters with optional user supplied prompting text.

PHOENIX EXAMPLES

(a) REQUEST PASSWORD

A line

***REQUEST PASSWORD=

is typed and the system waits for a value for the parameters PASSWORD to be typed on the same line.

(b) REQUEST 'PASSWORD= ' PASSWORD

When this is obeyed a message

PASSWORD=

is typed at the user's console and the system waits for the user to type a value on the same line.

(c) REQUEST 'PW? ' PASSWORD 'P? ' PROJECT 'PRINT? ' PRINTER$5K

The system prompts with

PW?
P?
PRINT?

waiting between lines for the symbolic parameter values to be typed. If no value is given for the PRINTER symbolic parameter, it takes the default value of 5K.

(d) REQUEST 'Highest return code to be ignored ' %RCLEVEL$0

The value of %RCLEVEL is set to the value typed by the user after the prompt, or to the default of 0 if no value is given.

REQUEST 2

PHOENIX COMMAND SYNTAX

```
     REQUEST  quoted-text[1] parameter
 or
     REQUEST  quoted-text[1] parameter$default
 or
     REQUEST  quoted-text[1] parameter quoted-text[1] parameter ...
 or
     REQUEST  quoted-text[1] parameter$default quoted-text[1] parameter$default ...
```

parameter may be either a symbolic parameter or a system variable.

NOTES

(a) A long prompt sequence may be short-circuited by giving a series of values separated by spaces on the same line.

(b) When a parameter is not given a value, the default, stripped of its quotes (if they exist) is used. If there is not default the parameter is left unchanged.

(c) If the REQUEST command is obeyed in a job, any default values are used to set the parameters.

(d) To give a parameter the value of the null string, the sequence '' must be used.

(e) The internal action of the REQUEST command is identical with that of the PARS command so that any of the usual values for the special parameters may be used.

---

1 Optional

RELATION

RELATION is a Phoenix command which may be used by the manager of a filespace to set up and delete authorised users (relatives) for the filespace. Authorised users of a filespace have the same powers of file protection as the manager of the filespace, except that they cannot themselves use RELATION

PHOENIX EXAMPLES

(a) RELATION SPQR FOR PUBLIB

SPQR is made an authorised user of the PUBLIB filespace.

(b) RELATION USER AB25 FOR SPQR DELETE

AB25 is deleted as an authorised user of SPQR's filespace.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
USER	text	-	-	Mandatory item
FOR	text	set name	-	-
-	text	CREATE	-	options
VER[5]	dsd	*	/N	-

OPTIONS

There are two options available. The default, CREATE, adds the user's name to the list of authorised users of the filespace. The second option is DELETE. This deletes the user named from the list of authorised users.

Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

RELEASE

RELEASE is a Phoenix command which may be used to release a 'saved' job (see SAVE). If the saved job is awaiting execution, the job is no longer marked as to be saved. If the job has executed and its output is 'saved' then it is queued for output to the appropriate destination. In order to release a job, the user must have the appropriate job queue access status (see PASSWORD).

PHOENIX EXAMPLES

(a) RELEASE 123

Job number 123 is released.

(b) RELEASE 4567 USER SPQR PW SECRET

JOB 4567 for user SPQR is released.

(c) JOB SPQR *,SAVE,PHX
....
UNLESS ERROR RELEASE

The job will be saved only if an error occurs; otherwise the output of the job will be printed.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
JOB	integer	current job	-
USER	text	value of U or USER	-
(PASSWORD / PW)	text	value of PASSWORD or PW	
VER[5]	dsd	*	/N

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

NOTES

(a) RELEASE cannot change the status of an executing job except as in (b)

(b) If RELEASE is obeyed in an executing job, which contains the SAVE command or JD parameter, and no job number is quoted, then the current job will not be saved.

RENAME

RENAME is a Phoenix command which names a temporary or anonymous dataset (the latter defined by a ddname) as a permanent dataset. An existing dataset of the same name will be lost if deletion is permitted by the protection status. RENAME should be used instead of FILE or COPY etc. whenever possible as it does not involve the copying of the contents of the dataset.

PHOENIX EXAMPLES

(a) RENAME &A AS .P

Temporary dataset is renamed as permanent dataset .P

(b) FTG1CLG (FT01F001 &TEMP/N) .....
UNLESS ERROR RENAME &TEMP AS .RES

If the Fortran programs runs successfully the output is kept as the permanent dataset .RES. If the program fails, the output to unit 1 is lost at the end of the job.

(c) C %H# (ERRS=%A)
.....
.....
.....
IF ERROR RENAME ERRS AS .ERRMESS
#

If an error occurs in the command sequence the anonymous dataset defined by the ddname ERRS is renamed as ERRMESS.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
FROM	dsd	-	-	Mandatory item
(AS / TO)	dsd	-	-	Mandatory item
VER[5]	dsd	*	/N	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted

RENAME
2

NOTES

If a ddname is used, it must refer to an anonymous dataset (%A). The anonymous or temporary dataset, as the case may be, is renamed with the specified permanent name. Any existing permanent dataset of the same name is deleted. The operation is done is such a way that it is very unlikely that both the old and the new dataset will be lost if the system crashes during the operation. The command produces no output unless it fails, in which case a message is sent to the VER dataset (defaulting to *) and a nonzero return code is given. If the operation fails, then the anonymous or temporary dataset remains in its original state. In order for the command to succeed the user must be able to create a new dataset of that name. The new dataset is given the creation status for the relevant filespace, but its contents and attributes are otherwise unaltered.

RETURN_CODES

If the command fails a nonzero return code is issued.

REQUEST

REQUEST is a Phoenix control command which is used to demand values of symbolic or special parameters with optional user-supplied prompting text. REQUESTG and REQUESTGLOBAL are synonyms for REQUEST. There is also a REQUESTLOCAL (synonym REQUESTL) command which may be used to demand values of local parameters.

PHOENIX EXAMPLES

(a) REQUEST PASSWORD

The line

```
***REQUEST PASSWORD=
```

is typed and the system waits for a value for the parameter PASSWORD to be typed on the same line.

(b) REQUEST 'PASSWORD=' PASSWORD

When this is obeyed a message

```
PASSWORD=
```

is typed at the user's console and the system waits for the user to type a value on the same line.

(c) REQUEST 'PW?' PASSWORD 'P?' PROJECT 'PRINT?' PRINTER$5K

The system prompts with

```
PW?
P?
PRINT?
```

waiting between lines for the symbolic parameter values to be typed. If no value is given for the PRINTER symbolic parameter, it takes the default value of 5K.

(d) REQUEST 'Highest return code to be ignored ' %RCLEVEL$0

The value of %RCLEVEL is set to the value typed by the user after the prompt, or to the default of 0 if no value is given.

(e) REQUEST 'Date?' DATE/REST$'Jan 1978'

or

REQUEST /REST 'Date?' DATE$'Jan 1978'

The value of DATE is set to the value given by the user or to the default of Jan 1978 if no value is given. The qualifier /REST allows the user to type in a value string containing spaces (see note (f)).

REQUEST
2

PHOENIX COMMAND SYNTAX

REQUEST [/qualifier][1] [quoted-text[1] parameter[/qualifier][1]][3]

or

REQUEST [quoted-text[1] parameter$default[/qualifier][1]][3]

parameter may be either a symbolic parameter or a system variable. See note (f) below.

NOTES

(a) A long prompt sequence may be shortened by giving a series of values separated by spaces on the same line.

(b) When a parameter is not given a value, the default, stripped of its quotes (if they exist), is used. If there is no default the parameter is left unchanged.

(c) If the REQUEST command is obeyed in a job, the parameters are set from the default values.

(d) To set a parameter to the null string, the sequence '' must be typed.

(e) The internal action of the REQUEST command is identical with that of the PARS command so that any of the usual values for the special parameters may be typed or used as the default.

(f) /REST (read rest of line) or /TEXT (read word or quoted string) are optional qualifiers. The default value is /TEXT.

---

1 Optional.
3 May be repeated.

REROUTE

REROUTE is a Phoenix command which may be used to reroute some or all of a job's output. Jobs which are inactive may be rerouted; executing jobs may only be rerouted from within the job. A job may be rerouted only if the user has the appropriate job queue access status.

PHOENIX EXAMPLES

(a) REROUTE 123 TO WESTCAM

The output for job 123 is rerouted to West Cambridge.

(b)
```
JOB SPQR *,PLOTTER 1K,*
...
...
IF ERROR REROUTE PLOT TO DUMMY
```

Plotter output is only produced if the job runs successfully.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
JOB	integer	current job	-	-
USER	text	value of USER or U	-	-
(PW / PASSWORD)	text	value of PW or PASSWORD	-	-
	(PRINT / PUNCH / PTAPE / PLOT / ALL)	ALL	-	-
TO[5]	text	-	-	mandatory ROUTE-name
VER[5]	dsd	*	/N	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

RETURN

RETURN is a Phoenix control command which causes a jump to be made, within the current command sequence, to a previously set up return label. RETURN values are saved and stored when using nested command sequences. See Section 4.10.4 for further details.

EXAMPLE

```
SET ERRJUMP ERRORS
SET RETURN YY
...
C .COM2
* the command sequence in .COM2 may
* contain  ERRJUMP and/or RETURN
....
*YY
SET RETURN  ZZ
....
QUIT
*ZZ
....
QUIT
*ERRORS
* Error routines
....
RETURN
```

If an error occurs and is not dealt with locally by use of an IF statement, and if the switch NOSTOP is unset, the error routine is entered. When it is complete, control returns to YY if the error occurred before the command SET RETURN ZZ, and to ZZ otherwise.

PHOENIX COMMAND SYNTAX

To set the return label

```
SET RETURN return label
```

To return to the label

```
RETURN
```

RUBOUT

RUBOUT is a Phoenix command which alters the effect of the rubout key and the escape combination @L.

PHOENIX EXAMPLES

(a) RUBOUT VDU

This sets the vdu rubout options.

(b) RUBOUT VERIFY

The verify rubout option is set.

(c) RUBOUT DEFAULT

The rubout option is reset to the default.

PHOENIX COMMAND SYNTAX

RUBOUT text

where text is one of VDU, VERIFY or DEFAULT

OPTIONS

DEFAULT	Rubout cancels the previous character and reflects '@C'. @L cancels the previous line and reflects '@L CR LF'.
VDU	Rubout cancels the previous character and reflects 'backspace space backspace', unless the cursor is already at the left hand side of the screen. Note that this does not give correct results after an escape combination as it only removes one character. It is also misleading after '@<return>' if characters before the '@<return>' are to be deleted. @L reflects 'CR spaces CR', the number of spaces being equal to the number of characters that were on the line. Note that this is misleading after '@<return>' as only the last part of the line is deleted.
VERIFY	Rubout cancels the previous character and reflects '\' followed by the character that was deleted. Successive rubouts continue to reflect deleted characters. When a character other than rubout is typed a delimiting '\' is typed. If rubout is typed when there are no more characters to delete, no reflection takes place after the initial '\'. If a character with EBCDIC code less than X'40' is typed, '?' is substituted. @L cancels the previous line and reflects '@L CR LF'.

RUN

RUN is a Phoenix command which may be used to submit a Phoenix job. The user's project number and any other JD parameters required must be set before the RUN command is used or with the command (see below). The Phoenix job may consist of a single command or a command sequence.

EXAMPLES

(a) PARS P=5678 U=ZYX1 PASSWORD=SECRET
PARS WHEN '10 MINS' LIMSTORE 32K
RUN PRINT .Z

generates and submits the following job:

```
JOB ZYX1 5678 PHOENIX JOB
ROUTE ,TURNROUND 10 MINS
PRIORITY ,POST ZXY1,PRINTER
TIME ,LIMSTORE 32K,PASSWORD SECRET
PHOENIX
PRINT .Z
```

(b) RUN FROM .COMSEQ JD PROJECT=1234 PRINTER 2K

The job is submitted with the project and printer limit set following the JD keyword, and the command sequence to be obeyed is read from the dataset .COMSEQ.

For a listing of the command sequence which submits the job type the dataset %SL:SUBRUN.

PHOENIX COMMAND SYNTAX

(a) RUN command

(b)

Key	Valuetype	Default	Note
FROM	dsd	%C	-
JD	parameter	-	see note (a) over declaration

N.B. The symbolic parameter JD may only be used when the FROM symbolic parameter is used implicitly or explicitly.

RUN
2

JOB DESCRIPTION PARAMETERS

Parameter	Alternative	Default	Note
9	-	SUBMIT	see note (b) below
USER	U	userid of logged on user	-
PROJECT	P or %PROJECT	-	must be set either locally or globally
PASSWORD	-	-	-
PRINTER	L or LINES		-
TITLE	-	PHOENIX JOB	-
ROUTE	R		-
POST	-		-
PRIORITY	Q		-
COMP	TIME		-
LIMSTORE	-		-
TURNROUN	W or WHEN		-
JD	-	-	-
JOBTYPE	JCL	PHOENIX	-
COMMAND	-	-	-

NOTES

(a) If any job description parameters are set on the RUN statement the parameter JD is mandatory.

(b) If the parameter 9 is set to TYPE before the RUN command is typed, the job is typed at a console and not submitted. To submit the job, set the parameter 9 to SUBMIT again; typing RUN now submits the job.

(c) The alternative parameters given above are implemented as defaults in the parameter-occurrences within the command sequence.
e.g. PRINTER <PRINTER$<L$<LINES>

RETURN CODES

A return code of 4 is set if a project number has not been set.

SAVE

SAVE is a Phoenix command which allows the output of an inactive job to be saved for later handling by other job queue commands (KILL, RELEASE, COLLECT, etc.). Both Phoenix and JCL jobs may be saved. In order to save a job a user must have the appropriate job queue access. It may be used in a job, in which case the job will save itself. (SAVE is also a job description parameter).

PHOENIX EXAMPLES

(a) SAVE 123

Job number 123 for the user identifier defined by the symbolic parameter USER or U will be saved after execution.

(b)
```
JOB SPQR *,*
  ...
  ...
  ...
IF ERROR SAVE
```

This job will be saved if an error occurred in the penultimate command.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
JOB	integer	current job	-
USER	text	value of U or USER	-
PW (PASSWORD)	text	value of PASSWORD or PW	-
VER	dsd	*	/N

NOTES

(a) Jobs which have been on the SAVE queue for some length of time (currently 24 hours) may be released by the system, but only if the job queue is almost full.

(b) All unit record (printer, punch or plotter) output is saved until the job is collected by the COLLECT command, killed by the KILL command or released by the RELEASE command or by the system.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Key may not be omitted.

SCRATCH

SCRATCH is a Phoenix command which deletes temporary datasets and ddnames This includes the current document and the previous current document if they are not currently open and the other system datasets. A temporary dataset name is not deleted if the dataset is currently open.

EXAMPLES

(a) SCRATCH T

all unallocated temporary datasets are deleted.

(b) SCRATCH D

all ddnames are deleted

PHOENIX COMMAND SYNTAX

SCRATCH options[1]

OPTIONS

The following options are available:

A deletes all temporary datasets and ddnames (this is the same as with no options supplied)

D deletes ddnames

T deletes all unallocated temporary datasets

---

1 optional

SEND

SEND is a Phoenix command and JCL procedure which sends messages to a Phoenix user If this user is logged on at the time a line

***MESSAGE FOR YOU

is typed at his console, otherwise the above line is typed during his next logon sequence See command MESSAGE for details of interrogation of the message

PHOENIX EXAMPLES

(a) SEND ABC1

sends the current document to user ABC1.

(b) SEND ABC1 A message

SEND ABC1 A message'

SEND ABC1 TEXT A message

each send the words 'A message' to ABC1.

(c) SEND ABC1 FROM .DOCUMENT

sends the contents of the file .DOCUMENT.

(d) SEND ABC1 FROM DOCUMENT TEXT 'FILE .DOCUMENT'

SEND ABC1 'FILE .DOCUMENT' FROM .DOCUMENT

each send the words 'FILE .DOCUMENT' followed by the contents of the file DOCUMENT

PHOENIX COMMAND SYNTAX

	Key	Valuetype	Default	Default Qualifier	Notes
(a)	SEND	-	userid	-	-	mandatory item Sends current document.
(b)	SEND	-	userid	-	-	mandatory item; must be first.
	TEXT	text / text in quotes	-	-	-

SEND
2

		Key	Valuetype	Default	Default Qualifier	Notes
(c)	SEND	-	userid	-	-	mandatory item. must be first
		FROM[5]	dsd	-	-	must be second
		TEXT[5]	text / text in quotes	-	-	-
(d)	SEND	-	userid	-	-	mandatory item. must be first
		TEXT	text in quotes	-	-	
		FROM[5]	dsd	-	-	-

all formats

STORE (integer / integer K) 36K

JCL EXAMPLES

(a) // EXEC SEND,TO=ABC1,TEXT='A MESSAGE'

sends the words 'A MESSAGE' to user ABC1.

(b) // EXEC SEND,FROM='AZY2.DOCUMENT',TO=ABC1

sends the contents of AZY2.DOCUMENT to user ABC1.

(c)
```
// EXEC SEND,FROM='AXY2.DOC',TO=ABC1,
//  TEXT='This is the interesting dataset'
```

sends the contents of AXY2.DOC preceded by the text to the user ABC1.

(d)
```
// EXEC SEND,TO=ABC1
//FROM DD *
   A long message ..........
//*
```

sends the message in the dataset defined by dd statement FROM.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
5 indicates that the key may not be omitted.
Keyed items may occur in any order; unkeyed items in the order given.

SEND
3

JCL PARAMETERS AND DDNAMES

The symbolic parameter TO must be present to identify the recipient of the message. The message to be sent may be given explicitly as the value of the symbolic parameter TEXT and/or implicitly by supplying a dd statement with ddname FROM.

NOTE

(a) If both the TEXT and FROM parameters are used the TEXT value is sent before the contents of the FROM dataset.

(b) There is a limit of about 1500 characters on the amount of MESSAGE text which may be stored up for any one user.

RETURN CODES

8 Message too long or MESSAGE file full.

12 MESSAGE file in use or user not a Phoenix user.

16 MESSAGE file broken or system error.

SET

SET is a Phoenix command which may be used to set up various values for later use. The values may be used for completing dataset names or to define record formats, space allocations etc. See 4.11.2 for further details.

PHOENIX COMMAND SYNTAX

	Key	Valuetype	Notes
SET	NAME[2]	text	Default if key missing
SET	CLEAR	-	see CLEAR
SET	(SW / SWITCH)	switchname	see SWITCH
SET	ERRJUMP	(label / *)	see ERRJUMP
SET	RETURN	label	see RETURN
SET	(P / PARS)	parameter declaration	see PARS
SET	TERMINAL	text	see TERMINAL
SET	CHAR	text	see CHAR
SET	(FORM / F)	text	see FORM
SET	(SPACE / SP / K)	text	see SPACE
SET	FD	text	see FD
SET	%Gn	text	see VARIABLES
SET	%Ln	text	see VARIABLES
SET	%QLEVEL	integer	see %
SET	%RCLEVEL	integer	see %

Choose one of the alternatives in brackets.
2 Key may be omitted.

SET
2

EXAMPLES

(a) SET RJS4

sets the set name to RJS4.

(b) SET PS93.FILE

sets the first part of a dataset name.

NOTES

(a) If SET NAME has not been used in a console session or job, the default value is the logged on user or the userid under which the job is being run.

(b) SET, when used to give a set name, also sets the variables %SET and the parameter S.

SETUP

SETUP is a Phoenix command which may be used to set up ddnames. Ddnames remain set up until the end of a console session or job or until the SCRATCH command is used or the ddname is deleted.

EXAMPLE

```
SETUP (FT10F001=.NEWFILE/N/FN DDNAM=*A)
```

The ddname FT10F001 is set up for a dataset .NEWFILE for use later in the job; DDNAM is set up to the printer or console.

NOTE

The SETUP command name is followed by a command environment (see Section 4.2).

SKIP

SKIP is a Phoenix control command which transfers control in a command sequence to the label specified. The label is defined by *label at the beginning of a line. SKIP may only be used for forward jumps; for backward jumps see JUMP. SKIP should always be used in preference to JUMP where possible, as it is much more efficient. Confusion may arise if a %H dataset containing the same label occurs in the command sequence. This statement may only be used within command sequences. See Section 4.10.4 for further details.

EXAMPLE

```
IF %RC LE 0 SKIP ON
QUIT
*ON
```

If the last command produced a condition code greater than 0, control returns to the calling level of commands, otherwise control is transferred to the label ON.

SNOBOL

SNOBOL is a text processing language with limited arithmetical facilities. It is available via Phoenix and JCL procedures, which can access the two compilers SNOBOL-4 and SPITBOL. SNOBOL-4 is the older compiler developed by Griswold et al. at the Bell Laboratories, and is a large (160K) load-and-go compiler. SPITBOL was developed by the Illinois Institute of Technology and can be used as a load-and-go or as a multistep compiler. It is appreciably smaller (120K) and faster than SNOBOL-4 and should therefore be used in preference. The procedures are called SNOBOL (for SNOBOL-4) and SPITBOL. Further details of the compilers may be found in the 'SNOBOL-4 Language Manual' and in the 'SPITBOL Version 2 Manual'

PHOENIX EXAMPLES

(a) SPITBOL PROGRAM=.PROG

compiles and runs a program stored in a catalogued dataset using all the default options.

(b)
```
SPITBOL (PROGRAM=%H$)  T=4
  A = INPUT
    . . .
  END
  $
```

compiles and runs a program from a %H dataset changing the default overall time limit from 1 to 4 seconds.

(c) SNOBOL PROGRAM=.PROG

compiles and runs the program using SNOBOL-4.

PHOENIX PARAMETERS AND DDNAMES

With both procedures, the program must be given using either the symbolic parameter or the ddname PROGRAM. The SPITBOL procedure also contains the parameters L, H, W, D, R, T, P and C which are used to change various store and time limits, and of which the most important is T which gives the overall time limit for compilation and running; so if a user needs a long running job, both the procedure time limit set by SECG, and T must have their values increased. It is good practice to make T less than SECG so that the SPITBOL compiler can gracefully terminate the program.

An object module may be produced by the SPITBOL compiler if a data definition using the ddname SYSOBJ is supplied in the environment. Further ddnames may be straightforwardly given in the environment for either input or output

Further details of the Phoenix parameters may be obtained by using the command HELP PROC SNOBOL and HELP PROC SPITBOL.

SNOBOL
2

JCL EXAMPLES

(a)
```
// EXEC SPITBOL
//SYSIN DD DSN=SPQR.PROG,DISP=SHR
```

compiles and runs from a catalogued disc dataset.

(b)
```
// EXEC SPITBOL,T=4
//SYSIN DD *
   A = INPUT


//*
```

The program is given in an instream dataset and the compile and run time is increased from 1 to 4 seconds.

(c)
```
// EXEC SNOBOL
//SYSIN DD DSN=SPQR.PROG,DISP=SHR
```

This is a SNOBOL-4 compile and run.

JCL PARAMETERS AND DDNAMES

The program must be defined with a dd statement with ddname SYSIN (or PROGRAM for SPITBOL or FT05F001 for SNOBOL-4). The SPITBOL JCL procedure has parameters with the same names as those in the Phoenix procedure (see above). Object modules may be produced if a dd statement SYSOBJ is supplied to the SPITBOL procedure.

Further details of the JCL parameters may be obtained by printing the ddatasets INFO.JCL.PARMS(SPITBOL) and INFO.JCL.PARMS(SNOBOL).

SORT-MERGE

SORT-MERGE is an IBM program package providing capabilities for users to arrange data into ordered sequences. It may be accessed via JCL catalogued procedures or Phoenix procedures. The package is fully documented in the IBM manual 'OS Sort-Merge Programmers Guide' (SC33-4007) and in the publication 'Use of the IBM Sort-Merge Program Product on the Cambridge IBM 370/165'.

PROCEDURES

The following are the procedures available:

SORT is for use when the linkage editor is required, and the user wishes to define his own work areas on tape or disc.

SORTB has seven pre-allocated disc datasets and is for use when the linkage editor is required but the user does not wish to define his own work datasets.

SORTC has seven pre-allocated disc datasets but none of the linkage editor datasets.

SORTD is for use when the linkage editor is not required, and the user wishes to define his work datasets.

It will be found that SORTC is in general the most useful procedure.

PHOENIX EXAMPLES

(a) SORTC (SYSIN=%H#) INPUT=SPQR.DATA SORTOUT=*

```
  Sort control statements
#
```

This is a simple sort of the contents of the catalogued dataset SPQR.DATA with the output from the sort going to the printer.

(b) SORTC (SYSIN=.CNTRLS) SORTOUT=.RES SORTIN=.DATA TRKS=60 MISC1=LIST

The input data, results and control statements are all set up from or to disc datasets. The three work space datasets have their sizes increased to 60 tracks and the MISC1 parameter is used to change the SORT-MERGE listing option.

SORT-MERGE 2

PHOENIX PARAMETERS AND DDNAMES

Primary input to the SORT-MERGE package must be defined using either the symbolic parameter INPUT or the ddname SORTIN. The size of disc work datasets may be changed from the default of 40 tracks by using the parameter TRKS and the listing and sorting options may be varied by using the parameter MISC1 (default value NOLIST) while the message-type is changed using the parameter MSG1 (default AP). The full list of parameters for each procedure may be obtained by using the command 'HELP PROC procedure-name', and a further discussion of the use of the parameters can be found in 'The Use of the IBM Sort-Merge Program Package on the Cambridge IBM 370/165'.

JCL EXAMPLES

(a)
```
// EXEC SORTC,SORTIN='SPQR DATA'
//SORT SYSIN DD *

 Sort control statements

//*
//SORTOUT DD SYSOUT=A
```

This is a simple sort of data contained in the disc dataset SPQR.DATA with the output from the sort being sent to the printer.

(b)
```
// EXEC SORTC,TRKS=60,MISC1=LIST,INPUT='SPQR.DATA'
//SORT SORTOUT DD DSN=SPQR.RES,DISP=(,CATLG),UNIT=DISC,
// SPACE=(TRK,(80,5))
//SORT SYSIN DD DSN=SPQR CNTRLS,DISP=SHR
```

The input, output and control statement datasets are all disc datasets in this example. The work dataset size is increased to 60 tracks by the use of the TRKS parameter and the MISC1 parameter is used to change the default listing option NOLIST to LIST.

Further examples may be found in Example Sheet 9.

JCL PARAMETERS AND DDNAMES

Primary input to the package is defined using either the symbolic parameter INPUT or by using an overriding dd statement with the ddname SORTIN In addition the dd statements with names SYSIN and SORTIN must be provided to define the control statement dataset and the output from the primary sort A full list of parameters may be found by printing the members of INFO JCL.PARMS with names corresponding to the procedure name of interest A detailed discussion of the use of the Sort-Merge package under JCL may be found in the manual 'Use of the Sort/Merge Program Package on the Cambridge IBM 370/165'.

COPYRIGHT

The Sort/Merge package is designated as proprietary software and users are referred to the warning in the introduction to Part II of this manual.

SP

SET SP is a Phoenix command which sets up a space qualifier for later use. SPACE and K are synonyms for SP.

EXAMPLES

(a)
```
SET SP1 5 2
INPUT TO .FILE/SP1
....
/*
```

The dataset .FILE is created with a primary allocation of 5 tracks and a secondary allocation of 2 tracks

(b)
```
SET SPACE2 3 1 6
FILE FROM .A TO .LIB:A/SP2
```

The pds .LIB is created with primary allocation 3 tracks, secondary allocation 1 track and 6 directory blocks

PHOENIX COMMAND SYNTAX

(a) SET (SPACE / SP / K) n m1 m2 m3

(b) SET SPn qualifier

n is the qualifier number and is an integer

m1 is the primary allocation in tracks and is an integer

m2 is the secondary allocation in tracks and is an integer

m3 is the number of directory blocks and is an integer

Either m3, or m3 and m2 may be omitted if a directory, or a directory and secondary allocation are not needed.

qualifier is any of the qualifiers defined in Section 4.3.3

NOTE

Normally one of the standard space qualifiers (SMALL, LARGE, MEDIUM, NORMAL, GIGANTIC) can be used, and so the SET SPACE command need only be used for repeated unusual allocations.

SPOOL

SPOOL is a Phoenix command applicable only in Phoenix jobs. It sets up a dataset to which output normally sent to COMMOUT0 and * datasets will be copied when the SPOOL switch is set. (The SPOOL switch is set automatically when the SPOOL command is obeyed). See SWITCH for details of the switch SPOOL. SPOOL is useful in Phoenix jobs for collecting together the output from several commands for later examination from a terminal.

EXAMPLE

```
SPOOL .RESULTS/VBA
```

PHOENIX COMMAND SYNTAX

```
SPOOL TO[1] dsd
```

NOTE

(a) The contents of the SPOOL dataset consists of the interleaved * and COMMOUT0 outputs. For example, if the QUIET switch is unset and the FTG1CLG procedure is obeyed then the spool dataset will consist of the reflection of the compilation command, followed by the output from the compiler, the reflection of the linkage edit command, the output from the linkage editor, the reflection of the command which runs the program and the output from the program.

(b) Output to datasets which have been defined by a SETUP command is not spooled.

---

1 Optional

SPSS

SPSS (Statistical Package for the Social Sciences) is an efficient package designed for fairly simple data manipulation and statistical analysis. There are two versions (SPSSG and SPSSH) which may be used via JCL catalogued or Phoenix procedures.

PROCEDURES

The following JCL catalogued and Phoenix procedures are available:-

SPSS	-	for normal use
SPSSX	-	for use only when one of the SORT CASES or ARCHIVE facilities is needed.

Note that the Phoenix procedures may not be used in a console session.

PHOENIX EXAMPLES

(a) SPSS FT05=.PROG

runs the package with control statements and data contained in the file .PROG.

(b) SPSS (SYSIN=%H+) PROG=SPSSH REGG=220K
.
.
+

runs the H-version with control statements and data contained in the %H dataset.

Further examples may be found in Example Sheet 18.

PHOENIX PARAMETERS AND DDNAMES

The control statements must be given using either the symbolic parameter FT05 or the ddname SYSIN. The version may be selected by the symbolic parameter PROG (default SPSSG) and the workspace altered by PARMG.

A complete list of parameters for each procedure may be obtained by using the command 'HELP PROC procedure-name'.

JCL EXAMPLES

(a) // EXEC SPSS,FT05='DSN=SPQR.PROG,DISP=SHR'

runs the package with control statements and data contained in the file SPQR.PROG.

SPSS
2

(b) // EXEC SPSS,PROG=SPSSH,REGG=220K
//SYSIN DD *

//*

runs the H-version with control statements and data contained in the instream dataset.

Further examples may be found in example sheet 18.

JCL PARAMETERS AND DDNAMES

The control statements must be given using either the symbolic parameter FT05 or the ddname SYSIN. The version may be selected by the symbolic parameter PROG (default SPSSG) and the workspace altered by PARMG.

A full list of parameters may be found from the partitioned dataset INFO.JCL.PARMS using the member-name which corresponds to the procedure name.

COPYRIGHT

The SPSS package is designated as proprietary software and users are referred to the warning in the introduction to Part II of this manual.

NOTES

SPSS is not the only statistical package available; others, such as GENSTAT, are more suitable for many applications. Users in doubt about the suitability of packages for particular problems should seek advice from the Consultancy Service.

REFERENCES

'Statistical Package for the Social Sciences', McGraw-Hill

Example Sheet 18.

STATUS

STATUS is a Phoenix command which interrogates the job queue.

EXAMPLES

(a) STATUS

Details of any jobs in the queue for the logged in user are typed.

(b) STATUS ABC1

A list of jobs in the queue for ABC1 is typed.

(c) STATUS ALL TO &A

The job queue is copied to a temporary file &A.

(d) STATUS PRINT255
or STATUS 'PRINT RECEPTION'

A list of jobs waiting to print on the Reception printers is typed.

(e) STATUS ,J801

A list of the jobs in the queue for the logged on userid and details of job number 801 are typed.

(f) STATUS WESTCAM

A list of jobs with route WESTCAM is typed.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
OF	text	logged on user	-	see below
TO[5]	dsd	*	/N	-
DATA[5]	dsd	$	/R	-

NOTES

(a) If a dataset is used to supply commands, lines will be truncated to 132 characters.

(b) When the STATUS command produces the message

PUNCH 0 PURGE

This means that the accounting information for the job has not yet been produced and the project's U and W values have not been updated for this job

---

Any item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Key may not be omitted.

STATUS
2

OPTIONS

The optional keyword OF may be followed by one or more of the following separated by commas:

user identifier

$\begin{pmatrix} n \\ Jn \end{pmatrix}$ where n is a job number.

A a list of all active jobs is typed.

ALL the first 25 jobs in the queue are output if output is to a console, otherwise as many jobs as possible, depending on the store available, are output.

JOBQ the number of jobs in the queue is output.

PRINT a list of jobs waiting to be printed is typed.

PRINTn the jobs waiting to print on printer n or at remote station n are typed.

Two particular values of n are

0 370 room printer

255 Reception

A list of current values of n may be obtained by using the command HELP JD ROUTE.

'PRINT name' the jobs waiting to be printed at the RJE called 'name' are typed.

CLASSn a list of jobs in the queue for that class is typed.

Rn a list of jobs in the queue with printer route set for remote station n.

PUNCH
PLOT a list of jobs waiting to punch or plot is output. At present, PUNCH and PLOT are synonymous.

PUNCHn
PLOTn a list of jobs waiting to punch or plot on punch n or plotter n is typed. PUNCH253 is the 370 room plotter.

Q may be used to terminate STATUS.

remote the jobs waiting to print at the named remote station.

The TO keyword specifies the destination of the output; the default value is *.

The DATA keyword defines a file from which further instructions will be taken after the text following the symbolic parameter OF has been interpreted.

RETURN CODES

STATUS issues a return code of 4 if there are no jobs in the queue for the user, and 8 for illegal options.

SUBMIT

SUBMIT is a JCL catalogued procedure and Phoenix command to submit one or more new jobs from a running job, or from a console, via the internal reader. The jobs may be in a dataset with any DCB characteristics, but any record longer than 80 characters (including those in instream datasets) will be truncated. The Phoenix command RUN may also be used to submit jobs.

PHOENIX EXAMPLES

(a) SUBMIT

A message giving the time of submission, priority etc.

```
 8 JUN 10.14 J0619SPQR PRIORITY 27.1 FAST
```

is output at the user's console, and the current document is submitted to the job queue.

(b) SUBMIT .A

.A is submitted, and a message similar to that above given.

(c) SUBMIT FROM %H!
....
!

This submits the instream dataset.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
FROM[1]	dsd	%C	/R

JCL EXAMPLES

(a) // EXEC SUBMIT,NAME='RJS4.JD'

This submits a job from a catalogued dataset and gives a message similar to that in Phoenix example (a) above.

---

1 Keyword is optional.

SUBMIT
2

(b)
```
// EXEC SUBMIT
//FROM DD DATA
JOB RJS4 444 NEW JOB
// . . .
. . .
. . .
/*
```

This submits from the instream dataset.

JCL PARAMETERS AND DDNAMES

The source dataset containing one or more complete jobs may be defined using the symbolic parameter NAME or by overriding the ddname FROM. There is no limit to the size or type of job that may be submitted. Note that if the job is submitted from an instream dataset this has to be introduced by a //FROM DD DATA statement, not DD *, and must be terminated by /*.

NOTE

The HASP terminator cards as defined in Section 2.3 can be useful in combination with the SUBMIT procedure for terminating jobs.

RETURN CODES

The following return codes may be issued

0 the job was submitted successfully.

4 the internal reader was busy, and the job was not submitted.

8 the internal reader was closed.

16 any other error; e.g. BREAK was pressed, there were too many lines in the file etc.

SUGGEST

SUGGEST is a Phoenix command which may be used to pass comments to Computing Service staff.

PHOENIX EXAMPLES

(a) SUGGEST The URM specification of suggest is unclear

The comment is stored in the system for later collection by Computing Service staff.

(b) SUGGEST FROM .TEXT

The message contained in .TEXT is added to the file of SUGGEST messages.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default
-	text	-
FROM[5]	dsd	%C

NOTES

This command is not intended to replace either the Comments Book or Forms, but merely to provide a convenient method for logged-on users to pass on comments.

Any key or item may be omitted unless otherwise stated.
5 Key may not be omitted.

SWITCH

SWITCH or SW is a Phoenix command which causes switches to be set and unset. Switches which may be set by the user are:-

COMMENT, ERRCODE, HOLDSP, NOEXEC, NOSTOP, NOSUB, NOSUBH, QUIET, REFLECT, RESTORE, SPOOL, TERSE, UPPER

There are also sixteen user switches; see User Switches.

SWITCH xxx, SW xxx and SET SW xxx are synonymous with SET SWITCH xxx.

PHOENIX EXAMPLES

(a) SWITCH QUIET

Set the QUIET switch.

(b) SET SW DUMP and SET -SW -DUMP

Both set the DUMP switch.

(c) SW -NOSTOP

Unset the NOSTOP switch.

(d) SW QUIET SW RESTORE

Set both switches.

PHOENIX COMMAND SYNTAX

(SW / SWITCH) (+ / - / NOT) name

SET (+ / -) (SW / SWITCH) (+ / - / NOT) name

Several switches may be set or unset by one command by repeating the keyword SW or SWITCH for each switch.

NOTE

Flags which are set by the system, but may be tested by the user, are:

ABEND, BADFILE, BADP, BADPARM, BADSYNT, ERROR, FALSE, FAST, OLDABEND, ONLINE, RC, TRUE

Specifications of these flags are listed separately in this chapter.

Choose one of the alternatives in brackets.

SWITCHES

Name	Action taken if switch set. The default value for each is unset unless stated below.
COMMENT	Any comment lines in command sequences are reflected if they begin with **, even if QUIET is set.
ERRCODE	If a command ends with a nonzero return code, the code is typed at the terminal.
HOLDSP	an area of main storage is reserved so that there will be enough room for abend processing. (Mainly for system use).
NOEXEC	commands are decoded and dataset allocations performed, but non control commands are not executed. NOEXEC is useful for syntax checking.
NOSTOP	the command stream continues to be obeyed after an error.
NOSUB	no substitution of parameters takes place in the current command sequence after the switch is set.
NOSUBH	parameter substitution in here datasets is suppressed at the level at which the switch is set.
QUIET	commands and messages are not reflected (but see COMMENT above and %).
REFLECT	each command is expanded into the full basic command and printed before execution begins.
RESTORE	when the current command sequence finishes all user-settable switches and the set string are restored to their settings on entry to this sequence.
SPOOL	all output which would otherwise be sent to a * dataset is sent to the dataset defined by a previous SPOOL command. See the SPOOL command.
TERSE	output from the compile and link edit steps of relevant procedures is produced in brief format.
UPPER	subsequent lines are forced into upper case before decoding by the command program. The switch affects the contents of here datasets, but not the contents of * datasets (e.g. INPUT). N.B. The switch does not affect the reflection of characters at a console.

TAPECOPY

TAPECOPY is a Phoenix command and a JCL catalogued procedure for copying one or more sequential datasets from one magnetic tape to another. The tapes may be either SL or NL, but when copying from an unlabelled (NL) tape to a labelled (SL) tape, the user must specify a FORM (see commands below). TAPECOPY adds the copied datasets to the end of the output tape unless otherwise directed by a WRITE command. The format of the output dataset is the same as that of the input dataset, although the DCB information in the label of the dataset may be altered. The name of the dataset may be changed, and a number optionally appended to the new dataset name. TAPECOPY is not suitable for copying one TLS tape to another TLS tape; see TLSCOPY.

PHOENIX EXAMPLES

(a)
```
JOB SPQR 1234
TAPE9 ABC101 XYZ202,PHOENIX
//TOTAPE DD VOL=SER=ABC101,UNIT=TAPE9
//FROMTAPE DD VOL=SER=XYZ202,UNIT=TAPE9
TAPECOPY
```

All datasets are copied to the output tape.

(b)
```
JOB SPQR 1234
TAPE9 ABC101 21XYZ3,PHOENIX
//TOTAPE DD VOL=SER=ABC101,UNIT=TAPE9
//FROMTAPE DD VOL=SER=21XYZ3,UNIT=TAPE9,LABEL=(,NL)
TAPECOPY %H! VER=&D
SET FORM 1 FB 80 800
COPY 1 AS SPQR.PROG/F 1
COPY 5 AS SPQR.RES/F 1
!
```

Two datasets are copied from an unlabelled to a labelled tape. Commands are taken from the instream dataset; reflected commands and error messages are filed to a temporary dataset.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
(WITH / SYSIN)	dsd	%N	/R	command stream
FROM	ddname	FROMTAPE	-	-
TO	ddname	TOTAPE	-	-
VER	dsd	*	/N	messages stream
STORE	(integer / integer K)	64K	-	-

Choose one of the alternatives in brackets.
Any item or key may be omitted unless otherwise stated.
Keyed items may appear in any order; unkeyed items in the order given.

TAPECOPY
2

JCL EXAMPLES

(a)
```
JOB SPQR 1234,TAPE9 ABC101 XYZ321,JCL
// EXEC TAPECOPY,TO=ABC101,FROM=XYZ321
```

All datasets are copied to the output tape.

(b)
```
JOB SPQR 1234,TAPE9 ABC102 99XYZA,JCL
// EXEC TAPECOPY,TO=ABC102,FROM=99XYZA,LABELF=NL
//SYSIN DD *
SET FORM 1 FB 80 800
COPY 1 AS SPQR.PROG/F 1
COPY 5 AS SPQR.RES/F 1
//*
```

Two datasets are copied from labelled to an unlabelled tape; commands are taken from the instream dataset.

JCL PARAMETERS AND DDNAMES

The tape to be copied is identified by a dd statement whose ddname is FROM or by the symbolic parameter FROM. TAPECOPY commands are read from a dataset identified by a dd statement with ddname SYSIN.

The following symbolic parameters are provided which define the input and output tapes:-

	From Tape	Default	To Tape	Default
Density:	DENF	3	DENT	3
Label type:	LABELF	SL	LABELT	SL
Unit:	UNITF	TAPE9	UNITT	TAPE9
Retention:	RETAINF	-	RETAINT	-
For 7-track only:	TRTCHF	-	TRTCHT	-

The destination tape is identified by a dd statement whose name is TO or by the symbolic parameter TO; the DCB parameters RECFM, BLKSIZE and LRECL, if included on this statement, are ignored. For a full list of the parameters available, print the file INFO.JCL.PARMS(TAPECOPY).

RETURN CODES

Error messages appear after reflected commands. If an error occurs within a file on the input tape, copying stops, unless the ERROR SKIP command is used. If any other I/O errors occur, processing is terminated, and an appropriate message printed.

The following return codes are issued:

0 successful run.
4 I/O or runtime error.
8 Command syntax or JCL error.

TAPECOPY
3

COMMANDS

Control statements are read from a dataset with symbolic parameter WITH or SYSIN (Phoenix only) or ddname SYSIN (JCL) which has RECFM=F or FB with LRECL=80 or RECFM=V or VB with actual record length not greater than 80 bytes.

Control statements are identified by keywords and are normally followed by one or more arguments separated by commas. Control statements are terminated by semicolon or end of record. At least one space is expected between arguments normally, but is not necessary when - / and & are used (see below and Note (a). The characters - * and & have special meanings. & is described in Note (a). - and UPTO are synonyms for TO when defining ranges of input dataset sequence numbers, e.g. 1-20, 1 TO 20 and 1 UPTO 20 are all equivalent. * means either the first or last dataset file on the tape, depending on the position.

e.g. *-20 is equivalent to 1-20;
5-* is equivalent to 5 TO the last dataset on the input tape;
- means the whole tape.

COPY COPY n1 COPY n1 TO n2 COPY AS args COPY n1 AS args COPY n1 TO n2 AS args	Defines the datasets to be copied, by specifying their sequence number. If no arguments are specified all the datasets are copied. See Note (a) for details of the AS clause. C is a synonym for COPY. n1 refers to the first dataset to be copied and n2 to the last.
ERROR STOP ERROR SKIP	Controls the action taken when a bad data block is READ. STOP terminates processing, SKIP skips over the bad block and processing continues. The default setting is ERROR STOP. ERR is a synonym for ERROR.
EXCLUDE n1 EXCLUDE n1 TO n2	Defines the datasets which are not to be copied. EX is a synonym for EXCLUDE.
EXPDATE DROP EXPDATE OFF	DROP removes expiry dates and should be used when copying TLS to SL tapes. OFF allows files on the output tape to be overwritten regardless of expiry date protection. At least one of DROP or OFF is required if EXPDATE is used. If both are used DROP must come before OFF.
RENAME n1 AS args RENAME n1 TO n2 AS args	Defines the datasets to be renamed. If n1 or n1 TO n2 is omitted all files processed so far are renamed. R is a synonym for RENAME. See Note (a) for details of the AS clause.
SET NAME a	Sets the default file owner. If no SET NAME command is issued, the user identifier under which the job was submitted is used. N and S are abbreviations for NAME and SET respectively.

TAPECOPY
4

SET FORM n1 recfm n2 n3	Defines the DCB information to be written into the tape labels of a dataset. n1 is the FORM identification number for an AS clause. recfm, n2 and n3 refer to the DCB subparameters RECFM, LRECL and BLKSIZE respectively. SET FORM is compulsory when copying from unlabelled tapes to labelled tapes. F is an abbreviation for FORM. See Note (a) for details of the AS clause.
TOTLS	The output tape is a TLS tape, TLSRECOV must be run subsequently with OPT=NR to recreate the directory.
WRITE FROM n1	Defines the tape sequence number, n1, on the output tape from which to start writing. If no write command is given, the last file on the output tape is located, and writing starts after that. W is an abbreviation for WRITE.

NOTES

(a) An AS clause is used to provide information for the dataset labels on the tape where none exist (on unlabelled tapes) or where the existing ones are to be overridden. An AS clause should always be used when copying unlabelled (NL) to labelled (SL) tapes. Note that an AS clause is only meaningful when the TO tape is labelled (SL).

Syntax: AS dsname&n1/FORM n2

or AS */FORM n2

dsname is the standard form of dataset name. If the dsname begins with a full stop, the current value of the SET NAME command (or if this has not been used the userid of the current job) is prefixed to the dsname. If dsname is omitted .FILE is used.

n1 is an integer appended to the dsname. Subsequent files have integers n1+1, n1+2 etc. If n1 is omitted the current value of n1 (or that dsname) is used. No appending takes place when &n1 is omitted.

n2 is the SET FORM number which defines the DCB characteristics for that file. If n2 is omitted 0 is assumed. If FORM n2 is omitted the DCB characteristics are copied from the FROM tape. An error occurs if the FROM tape is unlabelled.

* is used to indicate that the dsname is to be copied from the FROM tape but the DCB characteristics are to be altered.

(b) COPY and EXCLUDE are mutually exclusive.

(c) TOTLS and WRITE are mutually exclusive.

(d) WRITE and ERROR may appear only once in the control statements.

(e) FORMs and the SET name must be specified before they are used.

TAPECOPY
5

(f) RENAME must refer to files already defined explicitly (by COPY) or implicitly (by EXCLUDE).

(g) Unlabelled, standard labelled and TLS tapes are supported.

If the command stream dataset is absent 'COPY' is assumed.

Datasets are copied in sequence, therefore TAPECOPY commands must be in the order of the datasets on the FROM tape.

TAPESCAN

TAPESCAN is a Phoenix command and JCL catalogued procedure which scans any labelled or unlabelled tape, and prints the first few lines of each dataset.

If the wrong LABEL type is specified, the tape is reopened correctly and a message given.

If the dataset is an IEHMOVE pds, then a memberlist is given.

TLS tapes are treated specially - only the latest version of each non-deleted dataset is scanned. If the dataset is a pds, then a memberlist is given, otherwise the first few lines are printed.

PHOENIX EXAMPLES

(a)
```
JOB PC99 1234,TAPE9 PC9900
PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=PC9900
TAPESCAN
```

scans the standard labelled tape PC9900.

(b)
```
JOB PC99 1234,TAPE7 01PC99
PHOENIX
//TAPE DD UNIT=TAPE7,VOL=SER=01PC99,LABEL=(,NL),
//   DCB=(DEN=1,TRTCH=ET)
TAPESCAN OPT=X12/B
```

scans the 7 track unlabelled tape 01PC99
in BLP mode, printing the sample in Hexadecimal.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
OPT	text	-	-
TAPE	ddname	TAPE	-
OUTPUT[5]	dsd	*A	/N/VBA
STORE	(integer / integer K)	72K	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

TAPESCAN 2

JCL EXAMPLES

(a) JOB PC99 999,TAPE9 PC9900,JCL
// EXEC TAPESCAN,TAPE=PC9900

scans the standard labelled tape PC9900.

(b) JOB PC99 999,TAPE7 01PC99,JCL
// EXEC TAPESCAN,TAPE=01PC99,DEN=1,TRTCH=ET,
// UNIT=TAPE7,OPT='X12/B',LABEL=NL

scans the 7 track unlabelled tape 01PC99 as above.

JCL PARAMETERS

The symbolic parameter TAPE defines the tape volume to be scanned. All other parameters are optional.

LABEL=SL or NL	UNIT=TAPE9 or TAPE7
DEN=omitted, 1, 2, 3	TRTCH=omitted, C, E, T, ET

Default values are SL, TAPE9.

OPTIONS

Options are passed to the program by the OPT symbolic parameter or keyword. They may be separated by '/' , ',' or space(s). The default option is 'P7/X0/S1/E1/L0'. The options available are:

Pn	-	number of lines printed per file ($1 \leq n \leq 50$)
Sn	-	first file to start scanning at ($n \geq 1$)
Xn	-	print sample as a HEX dump of n lines ($1 \leq n \leq 50$)
T	-	truncate records to the width of the printer
W	-	continue even if wrong volume mounted (SL only)
B	-	treat as BLP
A	-	scan all versions of each dataset on the tape (TLS only)

the following apply to NL tapes, or those processed in BLP mode:

En	-	stop after n double tape marks ($n \geq 1$)

The following are for FB files on NL tapes or those processed in BLP mode only:

Ln	-	use this as LRECL if 'suitable' ($n \geq 1$)
N	-	do not attempt to interpret carriage control characters

'n' may be separated from the option letter by one or more spaces. If n=0, the option is treated as not being given.

Other illegal options are ignored - and an error given.

TAPESCAN
3

NOTES

(a) n consecutive tapemarks are treated as n-1 double tapemarks

(b) If both X and P options are given, the P option is ignored

(c) NL tapes (or those processed BLP) present a problem as there is no DCB information available to specify the format of the data

'VB' files will be distinguished from 'FB' files, but 'VBA will always be treated 'VB'

The blocking of 'FB' files has to be calculated from the size of the blocks and consequently some files may not be unblocked in exactly the correct format The value of the 'L' parameter is always used as the first 'LRECL , and if the blocksize is a multiple of it, the value of the L parameter is then used.

(d) 7 track tapes are not easily scanned by any program owing to the various ways in which the hardware can treat characters Consequently trouble (usually in the form of I/O errors) may be encountered when scanning 7 track tapes

RETURN CODES

Error messages are sent to SYSPRINT, if present, otherwise a return code of 999 is issued RCs greater than 4 cause scanning to be terminated The following are the return codes issued:

0	-	successful run
2	-	unknown option(s)
4	-	I/O error on a data block, or tape reopened.
8	-	I/O error on labels, incorrect labels etc.
12	-	no files on tape, insufficient REGION size or bad JCL
16	-	tape cannot be opened
999	-	error but unable to give message.

TAPESTAT

TAPESTAT is a Phoenix command and JCL catalogued procedure which may be used to find the access state, label type (NL or SL), rack number and date that the tape was last used. The various access states are described in Section 3.2.10. TAPESTATE is a synonym for the Phoenix command but not for the JCL catalogued procedure.

PHOENIX EXAMPLES

(a) TAPESTAT SPQR01

The label type, access state, location, rack number and date last used are output in the form:

```
SPQR01 SL WRITE ON REQUEST M/C ROOM 9999 27 MAY 76
```

(b) TAPESTATE TO &A SPQR02,SPQR03 05SPQR

Information for SPQR02,SPQR03 and 05SPQR is filed in &A.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
TO	dsd	*	/N	must be first
-	text	-	-	tape names

JCL EXAMPLES

(a)
```
// EXEC TAPESTAT,TAPE=SPQR01
```

The label type, access state, location, rack number and date the tape was last used are output, for example:

```
SPQR01 SL WRITE ON REQUEST M/C ROOM 9999 27 MAY 76
```

(b)
```
// EXEC TAPESTAT,TAPES='SPQR02,SPQR03 05SPQR'
//  PRINT1='&&A'
```

Information for SPQR02,SPQR03 and 05SPQR is filed in &&A.

Any key or item may be omitted unless otherwise stated.

TAPESTAT
2

JCL PARAMETERS AND DDNAMES

The names of the tapes to be examined are defined by the symbolic parameters TAPE or TAPES. Output from the program is defined by the symbolic parameter PRINT1, which goes to the printer by default.

NOTE

The list of tape names may be separated by either spaces or commas.

TERMINAL

The SET TERMINAL command is used to tell the communications system about the characteristics of a console which are different from the standard. Fixed lines are set up with the correct characteristics for the device. Options which are recognised on IBM 2741 terminals are marked *. For further details see Appendix D Sections 1, 2, and 8.

PHOENIX EXAMPLES

(a) SET TERMINAL WIDTH 60

This changes the terminal output width to 60 characters.

(b) SET TERMINAL NORUBOUT

The 'RUBOUT' character is ignored; @C may be used instead. This is useful on poor quality dial-up lines.

(c) SET TERMINAL SWCASE

The case of all subsequent letters is inverted. A full description of this is given in Appendix D, Section 8.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Equivalent Escape Combination	Notes
SET TERMINAL DEFAULT	-	-	@DD*	Reset to standard
SET TERMINAL DEPTH	integer	-	-	Not yet implemented
SET TERMINAL FULLCHAR	-	-	@DF	Full ASCII 96 character set
SET TERMINAL (HALFDUP / HALFDUPL)	-	-	@DH	Half duplex device
SET TERMINAL IDLEI	-	-	-	Carriage return as IBM 2741
SET TERMINAL IDLEN	-	-	-	Cancel previous SET TERMINAL IDLEs
SET TERMINAL IDLES	-	-	@DS	Slow carriage
SET TERMINAL IDLET	-	-	@DT	return
SET TERMINAL IDLEU	-	-	@DU	See App. D
SET TERMINAL IDLEV	-	-	@DV	Section 2

SET TERMINAL NOBREAK	-	-	@DB	'BREAK' ignored use @I
SET TERMINAL NOMESSAGE	-	-	-	Stop messages from being typed. See below.
SET TERMINAL NORUBOUT	-	-	@DR	'RUBOUT' ignored; use @C
SET TERMINAL RUBOUT	text	-	@DP & @DQ	see RUBOUT command.
SET TERMINAL (SWCASE / -SWCASE)	-	-	@DI * / @DJ *	Invert case See App.D Section 8.
SET TERMINAL WIDTH	integer	120 - IBM 2741 80 for certain VDU's 72 otherwise	-	-

NOTES

When SET TERMINAL NOMESSAGE is set all messages which come from any source other than the console session, e.g. operator messages, '***MESSAGE FOR YOU' etc. are suppressed.

TESTTERM

TESTTERM is a Phoenix command which can be run interactively under Phoenix to check out asynchronous terminals. It can generate a variety of character patterns, and it also has echoing facilities. This description assumes the use of the Phoenix Command Language. When testing an IBM 2741 terminal, for 'break' read 'attn' during output or @I during input.

EXAMPLE

```
TESTTERM
```

The program is entered.

NOTES

(a) Initialisation

Log on to Phoenix in the usual way.
Set the terminal width if necessary ('SET TERMINAL WIDTH n').
Enter the test program via the command 'TESTTERM'. It will type out a list of available tests (which may be stopped with 'break'), then enter its control state.

(b) Selecting a test

When the program is in control state, it prompts with a hash sign (#) for a command to select another test. Commands consist of one or two unsigned decimal numbers, separated by one or more spaces or tabs. The first number specifies which test is to be run, the second number is a count (referred to as 'n' below), which is used in different ways by different tests. The default value is 0 which is interpreted as infinity. A test may be abandoned at any time by pressing 'break'.

(c) Available tests

In all the tests which type n lines of a particular character string each line begins one character later in the string.

0. Type a list of tests n times.

1. Full EBCDIC. Type n full lines of Cambridge defined printing characters in EBCDIC collating sequence. Not all terminals will be capable of printing all the characters; those that are not available will be converted to '?'. The character string is as follows:-

```
[]{} £.<+|&!$*);¬-/\vΛ↑←⅋,%_>?`:#@'="
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
```

2. Special characters. Type all the defined special characters n times. A separate line is used for each character, which is preceded by its name and enclosed in brackets. The special characters are:-

NUL TAB DEL VT FF CR SO SI NL BS IGS IUS LF ESC BEL SP

TESTTERM
2

3. Direct Echo. Echo n lines of input. The program prompts with a dot (.) for each line to be echoed; output is indented with one space to align it with the input. An input line which is longer than the current terminal width will be echoed as two lines. Break can be used to end this text before n lines have been reflected, and must be used if n=0 (infinity).

4. Hexadecimal Echo. Echo n lines of input as the hexadecimal values of the EBCDIC characters received.

5. 2741 Shift Test. Type n full lines of the IBM 2741 character set, with the characters in such an order as to maximize the rotations of the type head.

6. Triangular EBCDIC. Type n lines of length 1,2,3...etc characters from the EBCDIC printing character set.

7. Repeated letters and digits Type n full lines of characters from the following string:-

AAABBBCCC...XXXYYYZZZ000111222...777888999aaabbbccc...xxxyyyzzz

8. Triangular repeated letters and digits. Type lines of length 1,2,3.. etc from the character set of test 7

9. Specified line repeat. Read one line of input and type it n times. If a line longer than the terminal width is input, the excess characters are ignored.

10. 2741 Carriage Functions Test Types a series of tabs, spaces, backspaces and linefeeds, saying what it is doing.

(d) Stopping the program

To stop the program, type 'Q' instead of a test number. The return code from this program is always zero

(e) Errors

After all errors, the program reverts to its control state and prompts for another control input line

If a character other than a space, tab or digit is encountered in control input, the program types a plus (+) underneath the offending character, and then gives the message "non-digit (X'<hex value>')". If a number greater than 2147483647 is read, the program gives the message "number too large". If a non-existent test is called for, the message "no test <n>" is typed

If more than two numbers are read in a line of control input, the program gives the message "more than two numbers read".

TLS

The Tape Library System (TLS) comprises a set of programs providing comprehensive facilities for maintaining a library of files on a magnetic tape. The programs are TLS, the main file manipulating program; TLSINIT, the tape initialising program; TLSCOPY, which moves files from one TLS tape to another; and TLSRECOV which is a program for reconstructing the dictionary of a tape after its loss due to I/O errors.

STRUCTURE OF A TLS TAPE

TLS requires a dedicated 9-track tape, with standard labels. All files on the tape have standard IBM labels and are expiry date protected. A tape may contain up to 186 files. The first file is reserved for the dictionary which holds control information related to the current contents of the tape; the dictionary is maintained by the TLS procedures and is not accessible to the user. Other files on the tape can be read by any program, independently of TLS, provided that they have DSORG=PS, as shown by the TLS EXAMINE listing. Files that have DSORG=PO cannot be processed by any program other than TLS. Under no circumstances should files be written to a TLS tape by means of any programs other than TLS or TAPECOPY's special TOTLS option (see TAPECOPY). The trailer labels for each file contain a copy of the directory entry for the file, enabling the directory to be reconstructed should a tape failure occur on reading it

INITIALISING THE TAPE - TLSINIT

Tapes to be used in conjunction with TLS must be initialised by the program TLSINIT This is in addition to the standard initialisation for labelled tapes which is performed automatically by the Magnetic Tape Librarian. Magnetic tapes will be initialised for TLS by the Tape Librarian on request, when the tape is issued.

TLSINIT should never be run again unless all datasets on the tape are to be lost.

TLS
2

TLS

TLS is a Phoenix command and JCL catalogued procedure which may be used to maintain a library of datasets on a magnetic tape. See preceding page for details of TLS tapes.

PHOENIX EXAMPLES

(a) JOB URM9 123,TAPE9 URM901

```
//TAPE DD UNIT=TAPE9,VOL=SER=URM901
TLS WITH %H!
TT .FILE
EX
!
```

URM9.FILE is moved to tape URM901 and a list of the current versions of all non-deleted datasets is printed.

(b) JOB URM9 123,TAPE9 URM901

```
//TAPE DD UNIT=TAPE9,VOL=SER=URM901
TLS WITH %H! VER=.B
  commands
!
```

TLS messages are filed to URM9.B.

For further examples see Example sheet 21.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
(WITH / SYSIN)	dsd	-	/R	command stream mandatory item
TAPE	ddname	TAPE	-	-
VER	dsd	*	/N	message stream
OPT	text	-	-	provided for future development
STORE (integer / integer K)	78K	-	-	
ENV	text	-	-	extra data definitions

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

TLS
3

JCL EXAMPLES

(a)
```
JOB URM9 123,TAPE9 URM901,JCL
// EXEC TLS,TAPE=URM901
//SYSIN DD *
TT .FILE
//*
```

URM9.FILE is moved to tape.

(b)
```
JOB ABC1 999,TAPE9 ABC101,JCL
// EXEC TLS,TAPE=ABC101
//DD1 DD DSN=ABC1.DATA(A),DISP=SHR
//SYSIN DD *
TT DD1 AS .DATA
EX FIFO
//*
```

A member of the partitioned dataset ABC1.DATA is moved to tape as a sequential dataset. All non-deleted datasets on the tape are printed in a 'first in first out' order.

See Example Sheet 21 for more examples.

JCL PARAMETERS

The TLS tape is defined by the symbolic parameter TAPE.

For a full list of the parameters available print INFO.JCL.PARMS(TLS).

COMMANDS

Each command is identified by one or exceptionally two keywords, which may be followed by one or more sets of arguments separated by commas. Commands may be given in upper or lower case and are terminated by semi-colon or end of line (except within a member list, when end of line is ignored after a comma). Command lines may be up to 120 characters long. If no commands are given, the tape dictionary is examined (equivalent to the command EX). Comments may be given in the command stream. A comment is introduced by the character '\' or '¦' and is terminated by the end of line.

CHANGE x 'remarks' CHANGE x(-n) 'remarks'	The remarks field (as defined below - see TT) is altered. x is a tape sequence number, or a dataset name optionally followed by a version number, -n, enclosed in brackets (see note (h)). CHA is a synonym for CHANGE.
DELETE x DELETE x(-n)	Mark all versions, or a single version, of a dataset deleted. x is a tape sequence number, or a dataset name optionally followed by a version number, -n, enclosed in brackets (see note (c)). If x is a dataset name and is not followed by a version number, all

TLS
4

	versions of the dataset are deleted, otherwise only the specific version is deleted. A deleted version of a dataset is not available by name in any commands, but may be referenced by tape sequence number. Deleted versions of datasets are skipped in the EXAMINE listings, unless ALL is specified. DEL is a synonym for DELETE. See note (h).
ECHO	Reflect subsequent command lines.
ERROR NOSTOP	Continue processing even if there are command errors.
ERROR [STOP] n	Stop after command analysis if there are command errors of level n or greater. (The key STOP is optional.) The action of ERROR 0 is always to stop after command analysis. The default is ERROR STOP 8.
EXAMINE EXAMINE -n EXAMINE ALL	Prints the dataset information held in the dictionary. The entries are printed in alphabetical order. The first command prints the most recent version of all non-deleted entries, the second prints all non-deleted entries up to and including version -n (see note (c)). The d prints all entries (including deleted entries, which are marked by an asterisk) in alphabetical order. Only one of these commands may be specified in any call of TLS. See note (g) for details of the output. EX is a synonym for EXAMINE.
EXAMINE FIFO EXAMINE FIFO n EXAMINE FIFO ALL EXAMINE LIFO EXAMINE LIFO n EXAMINE LIFO ALL	Entries which refer to non-deleted datasets are printed in the order in which they appear on the tape; first in first out if LIFO is specified and last in first out if LIFO is specified. If n is specified, only the last n entries are printed. If ALL is specified, all entries are printed, deleted files being marked by an asterisk. Only one of these commands may be requested in any call of TLS. See note (g) for details of the output. EX is a synonym for EXAMINE and SEQ is a synonym for FIFO.
FILEDUMP x FILEDUMP x 'remarks'	Transfer a single dataset or group of datasets to tape if the dataset(s) are not already on the tape. x may be a dataset name or a dataset index name; if omitted, the current SET name is used. If x is a dataset index name, all the datasets that start with the component(s) x are checked for transfer. The remarks string (as defined below - see TT), if given, is associated with each dataset transferred to tape. All the

TLS
5

	datasets checked by the FILEDUMP command are listed in the TO TAPE section of the TLS output and are marked by the characters '(+)'. ARCHIVE, FD and ARC are synonyms.
FT x FT x(-n) FT x(list) FT x(-n)(list) FT x AS d FT x(-n) AS d FT x(list) AS d FT x(-n)(list) AS d	Transfer a dataset from tape. x may be a tape sequence number, or a dataset name optionally followed by a version number, -n, enclosed in brackets (see note (c)). If the dataset is partitioned, a list of members may be specified (see note (i)). The dataset is transferred to a dataset of the same name on disc unless the optional AS clause is specified, in which d may be a ddname or a dataset name. If the disc dataset already exists and is partitioned, the tape pds is merged into the disc pds, provided that the DCB information of the datasets is compatible. If there is a member of the same name in both datasets, the member from the tape pds replaces the member in the disc pds. If the disc dataset is specified on a dd statement, the SPACE information should be based on the information in the dictionary entry for the tape dataset (see note (g)) and the DCB information should be omitted. If a single membername and an AS clause are specified, a change of DSORG is attempted, see note (j). FT is an abbreviation for FROM TAPE.
NOECHO	Do not reflect subsequent command lines (unless there is an error). The default is NOECHO. QUIET is a synonym for NOECHO.
RECOVER x	Recover a deleted entry. x may be a dataset name or tape sequence number. All versions of the dataset are restored to the non-deleted state if x is a dataset name, otherwise only the specific version. REC is a synonym for RECOVER. See note (h).
SET x	Any dataset name contained in the command stream which begins with a fullstop will be prefixed by x. x is text which consists of one or two dataset name components. If no SET command is given, the identifier under which the job is being run, or the SET NAME value (Phoenix only) is assumed.
TT x TT x(list) TT x AS d TT x(list) AS d TT x 'remarks' TT x(list) 'remarks' TT x AS d 'remarks' TT x(list) AS d 'remarks'	Transfer a dataset to tape. x may be a dataset name or a ddname referring to a dd statement defining the dataset, and may be followed by a list of members if the dataset is partitioned (see note (i)). The AS clause is optional for a permanent disc file and mandatory for a temporary file; if it is included, the dataset has the specified dataset name, d, on the tape, otherwise the dataset name is the same as the disc dataset

TLS
6

name. The command may be terminated by a remarks string of up to 24 characters, which will be printed whenever the dictionary is examined. The characters are introduced by a quote and terminated either by another quote or end of line - semicolon does not terminate a remarks string. All characters are converted to upper case. If a quote character is to be included, two consecutive quotes must be coded. If a single membername and ar AS clause are specified, a change of DSORG is attempted, see note (j). TT is an abbreviation for TO TAPE.

NOTES

(a) Datasets may only reside on disc or tape.

(b) Dd statements are unnecessary for permanent disc datasets. Disc datasets may be referred to directly by name or by the ddname on which they are specified. Temporary datasets need only be specified on dd statements if TLS is being run under JCL. When running under Phoenix, temporary datasets can be created dynamically, but as the Phoenix dataset &T and the JCL dataset &T do not refer to the same dataset, dynamic creation of temporary datasets is not allowed under JCL.

(c) When a tape dataset is created it is entered in the dictionary and is known as version 0 of the dataset. When a new dataset with the same name is subsequently added to the tape, this new dataset is entered in the dictionary as version 0 and the earlier version becomes version -1. If the process is repeated, the earliest version becomes version -2 etc. Any version of the dataset may be moved to disc by reference to the appropriate version number. If a version is deleted, all previous versions decrease their version number by 1, and the reverse applies when a version is recovered.

(d) Irrespective of the order in which they appear in the command stream, all FROM TAPE commands are executed before any TO TAPE commands. DELETE, RECOVER and SET commands are executed as they are read. The listing requested by means of the EXAMINE command is produced after all other commands have been obeyed.

(e) The tape sequence number refers to the order of the datasets on the tape. In an EXAMINE listing this is the number printed on the extreme left hand side. The first dataset placed on the tape by TLS is sequence number 2. If the dataset is to be read directly by a user program, the sequence number should be used in the LABEL field of the dd statement.

(f) The dictionary may be saved on disc after a run of TLS provided it has been changed. If ddname TLS05 exists and references a disc dataset, the dictionary is written out. Subsequently, it may be read by TLSEX or TLSRECOV if the tape copy of the dictionary fails.

TLS
7

(g) When EXAMINE is used the following information is printed for each dataset:-

Tape sequence number - see note (e). An asterisk in front of the number indicates that the version has been deleted.

Dataset name.

Dataset format; DSORG, RECFM, LRECL and BLKSIZE.

Dataset size in tracks; datasets with DSORG=PO have the size followed by the number of blocks required for the pds directory. The size given is the amount of space required for moving the dataset back to disc.

Date when dataset was copied to tape and date when the disc dataset was last modified prior to that.

Number of blocks in the dataset (sequential only).

Remarks.

(h) If the dictionary has to be recovered by TLSRECOV, all CHANGE, DELETE and RECOVER commands that have been used prior to the recovery will be ignored and all datasets will appear in their initial state.

(i) When a partitioned dataset is specified for a TT or FT transfer, all members are transferred unless a list is specified. The list has two forms:

```
(membername,membername,.....)
(EXCEPT membername,membername,.....)
```

In the first form, only those members specified are transferred. In the second case all members BUT those specified are moved. New line is ignored within memberlists, when preceded by a comma.

(j) A single member of a pds may be transferred to or from tape as a sequential dataset, or a sequential dataset as a single member of a pds, if the membername and an AS clause are specified. This change of DSORG will be accomplished provided no pds directory information is lost, or no conflict of DSORG occurs. When a single membername is specified, the member is logically treated as a sequential dataset.

TLS
8

RETURN_CODES

Only the highest return code encountered is given. The following return codes are issued:-

0	-	successful run
4	-	command error of level 4 e.g. Overlong remarks Invalid membername in list
8	-	command error of level 8 e.g. Unknown command
12	-	one or more FT transfers failed
16	-	one or more TT transfers failed
20	-	tape full i.e. End of tape reached Dictionary full
32	-	dictionary could not be written to tape
999	-	dictionary could not be read
	-	tape is not a TLS tape
	-	insufficient REGION
	-	other serious failures

TLSCOPY

TLSCOPY is a Phoenix command and JCL catalogued procedure for copying all or part of a TLS tape to another TLS tape. The datasets that are copied are added to the end of the output tape. Datasets to be copied can be selected by various commands which supplement the default action of copying the most recent version (version 0) of all non-deleted datasets.

PHOENIX EXAMPLES

(a)
```
JOB SPQR 567,TAPE9 ABC999 DEF111
PHOENIX
//TOTAPE DD UNIT=TAPE9,VOL=SER=ABC999
//FROMTAPE DD UNIT=TAPE9,VOL=SER=DEF111
TLSCOPY
```

This copies the most recent version of all non-deleted datasets.

(b)
```
JOB SPQR 123,TAPE9 RJS971 PS1910
PHOENIX
//TOTAPE DD UNIT=TAPE9,VOL=SER=RJS971
//FROMTAPE DD UNIT=TAPE9,VOL=SER=PS1910
TLSCOPY WITH %H!
  commands
!
```

The tape is copied according to the command stream.

See Example sheet 21 for more examples.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
(WITH / SYSIN)	dsd	$	/R	command stream
FROM	ddname	FROMTAPE	-	-
TO	ddname	TOTAPE	-	-
VER	dsd	*	/N	messages stream
DSN	dsd	$	/R	disc dataset
OPT	text	-	-	option list
STORE	(integer / integer K)	64K	-	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

TLSCOPY 2

JCL EXAMPLES

(a)
```
JOB SPQR 567,TAPE9 ABC101 XYZ999,JCL
// EXEC TLSCOPY,TO=ABC101,FROM=XYZ99
```

copies the most recent version of all non-deleted datasets.

(b)
```
JOB SPQR 123,TAPE9 IJK567 STU999,JCL
// EXEC TLSCOPY,TO=IJK567,FROM=STU999
//SYSIN DD *
  commands
//*
```

This copies the tape according to the command stream.

JCL PARAMETERS AND DDNAMES

The input and output tapes are defined by the symbolic parameters FROM and TO respectively or by the ddnames TLS11 and TLS01 respectively. Options are passed to the program via the symbolic parameter OPT, and if the dictionary is to be read from disc, this should be defined by the DSN symbolic parameter.

Printed output is sent to a dataset defined by PRINT1.

For a full list of symbolic parameters print the file INFO.JCL.PARMS(TLSCOPY).

OPTIONS

Options are passed to the program by the OPT symbolic parameter, and are separated by spaces, commas or /. The default value is S4. The options available are:-

D — specifies that the dictionary of the input tape be read from the disc dataset (identified by ddname TLS04) and not from the tape. The dictionary would have been placed there by a previous run of TLSRECOV.

Sn — where n > 3 (unless option L specified) specifies the absolute (BLP) dataset that contains the header labels of the first (BLP) user dataset on the input tape that has a valid entry in the dictionary.

L — specifies that the dictionary area of the input tape should not be verified. This causes the D option to be set implicitly and the input tape to be opened BLP. There is no numeric argument.

These options correspond to the options in TLSRECOV and are provided to allow a tape that has a lost or corrupted dictionary to be copied to another tape.

TLSCOPY
3

COMMANDS

Each command is identified by a keyword, which may be followed by one or more sets of arguments separated by commas. Commands may be given in upper or lower case and are terminated by semi-colon or end of line. Command lines may be up to 120 characters long. If no commands are given, the most recent version of all non-deleted datasets is copied (equivalent to the command COPY). Comments may be given in the command stream. A comment is introduced by the character '\' or '¦' and is terminated by the end of the line.

COPY COPY 'remarks'	Copy the most recent version of all non-deleted datasets. The remarks string, if given, is associated with every dataset that is copied.
COPY ALL	Copy all versions of all datasets, whether deleted or not. This command overrides any COPY, MERGE or EXCLUDE commands. Remarks strings may be changed and datasets renamed by use of appropriate commands. See note (f).
COPY NODEL	Copy all non-deleted/excluded versions of all datasets. This command overrides all COPY or MERGE commands. Remarks strings may be changed and datasets renamed by use of appropriate commands. NOEXC is a synonymn for NODEL. See note (f).
COPY x COPY x(-n) COPY x 'remarks' COPY x(-n) 'remarks' COPY x AS y COPY x(-n) AS y COPY x(-n) AS y 'remarks' COPY i TO j COPY i TO j 'remarks'	Copy the specified dataset. Use of the COPY command overrides the default COPY action. x may be a tape sequence number, or a dataset name optionally followed by a version number, -n, enclosed in brackets. The dataset name may be changed by specifying an AS clause, and a new remarks string may also be given. In the final form of the command, a sequence of datasets between and including the tape sequence numbers i and j are copied. The remarks string, if given, is associated with each dataset copied.
CHANGE x 'remarks' CHANGE x(-n) 'remarks'	The remarks field of the dataset is altered. x is a tape sequence number, or a dataset name optionally followed by a version number, -n, enclosed in brackets. This command does not automatically cause the dataset to be copied. CHA is a synonym for CHANGE.
ECHO	Reflect subsequent command lines.
ERROR NOSTOP	Continue proce ing even if there are command errors.

TLSCOPY
4

ERROR [STOP] n	Stop after command analysis if there are command errors of level n or greater. (The key STOP is optional). The action of ERROR 0 is always to stop after command analysis. The default is ERROR STOP 8.
EXAMINE	Print the dataset information held in the dictionary of the output tape. See the main TLS specification for the details of the syntax. EX is a synonym for EXAMINE.
EXCLUDE x EXCLUDE x(-n) EXCLUDE i TO j	Do not copy a dataset or group of datasets. x may be a tape sequence number, or a dataset name optionally followed by a version number, -n, enclosed in brackets. In the final form of the command, a sequence of datasets between and including the tape sequence numbers i and j are not copied. EXC is a synonym for EXCLUDE. See note (e).
MERGE x MERGE x 'remarks'	Copy a dataset or group of datasets if the dataset(s) do not already exist on the output tape. x may be a dataset name or a dataset index name, in which case the latest version of each non-deleted dataset that starts with the component(s) x is checked for copying. If x is omitted, the latest version of all non-deleted datasets is checked for copying. The remarks string, if given, is associated with every dataset that is copied. All datasets that are checked for copying by the MERGE command are listed in the TLSCOPY output and are marked by the characters '(+)'. Use of the MERGE command overrides the default COPY action. MER is a synonym for MERGE.
NOECHO	Do not reflect subsequent command lines (unless there is an error). The default is NOECHO. QUIET is a synonym for NOECHO.
RENAME x AS y RENAME x(-n) AS y RENAME x AS y 'remarks' RENAME x(-n) AS y 'remarks'	Rename the specified dataset and optionally alter its remarks string. x may be a tape sequence number, or a dataset name optionally followed by a version number, -n, enclosed in brackets. y is the new dataset name. This command does not automatically cause the dataset to be copied. RN is a synonym for RENAME.
SET x	Any dataset name contained in the command stream which begins with a fullstop will be prefixed by x. x is text which consists of one or two comp[illegible]t names. If no SET command is given the identifier under which the job is being ran, or the SET NAME value (Phoenix only) is assumed.

TLSCOPY
5

NOTES

(a) Commands usually refer to the most recent version (0) of a dataset. Other versions may be referenced by specifying the relevent version number in brackets after the dataset name, or by specifying the tape sequence number. Datasets appear as version 0 on the output tape when they are copied and if more than one version of a dataset is copied to the output tape, the relative sequence of versions is preserved.

(b) Irrespective of the order of commands, datasets are copied in input tape sequence number order. The listing requested by means of the EXAMINE command is produced after the copying has finished.

(c) The commands EXCLUDE, CHANGE and RENAME are primarily provided to supplement the COPY and MERGE commands or the default (implicit) COPY command. See also note (e).

(d) Specifying any COPY command or MERGE command overrides the default (implicit) COPY command, but this can be specified explicitly at any time. COPY is more binding than MERGE.

(e) The EXCLUDE command is equivalent to the DELETE command in TLS. Excluding a dataset (or a version of a dataset) with TLSCOPY has essentially the same effect as deleting it with TLS immediately before the run of TLSCOPY and recovering it with TLS after the run. A deleted or excluded version of a dataset may be copied by referring to it by tape sequence number either explicitly, or implicitly within a COPY range.

(f) Dummy entries (those with dataset names beginning '...') which are created by TLSRECOV, to account for lost datasets on a corrupted tape, are always bypassed by TLSCOPY, even if 'COPY ALL' or 'COPY NODEL' is specified.

(g) The dictionary of the output tape may be saved on disc after a run of TLSCOPY, provided that is has changed. If ddname TLS05 exists and references a disc dataset, the dictionary is written out. It may be subsequently read by TLSEX, or TLSRECOV if the tape copy of the dictionary fails.

TLSCOPY
6

RETURN_CODES

The following return codes are issued. Only the highest return code is given.

0	-	successful run
2	-	unknown options
4	-	command error of level 4 e.g. Overlong remarks
8	-	command error of level 8 e.g. Unknown command
16	-	one or more dataset copy failures
20	-	output tape full i.e. End of tape reached Dictionary full
32	-	output dictionary could not be written to tape
999	-	either dictionary could not be read
	-	either tape is not a TLS tape
	-	insufficient REGION
	-	other serious failures

TLSINIT

TLSINIT is a Phoenix command and JCL catalogued procedure which must be used to initialise (the dictionary dataset is written) a TLS tape before any files may be written to the tape by TLS. Tapes which are to be used by TLS will be initialised by the Magnetic Tape Librarian when the tapes are issued, if requested. TLSINIT should never be used again unless all TLS files on the tape are to be deleted. TLSINIT may also be used to scratch a TLS tape, initialising the tape as a non-TLS standard labelled tape.

PHOENIX EXAMPLES

(a)
```
JOB SPQR 999,TAPE9 SPQR01
PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=SPQR01
TLSINIT
```

This job initialises the tape SPQR01.

(b)
```
JOB SPQR 999,TAPE9 SPQR02
PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=SPQR02
TLSINIT OPT D800
```

This initialises the tape at 800 bpi.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
OPT	text	D1600	-	-
TAPE	ddname	TAPE	-	-
VER	dsd	*	/N	message dataset
STORE	(integer / integer K)	16K	-	-

JCL EXAMPLES

(a)
```
JOB SPQR 999,TAPE9 SPQR01,JCL
// EXEC TLSINIT,TAPE=SPQR01
```

This job initialises the tape SPQR01.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

TLSINIT
2

(b) JOB SPQR 999,TAPE9 SPQR02,JCL
// EXEC TLSINIT,TAPE=SPQR02,OPT='D800',RETAIN=RETAIN

The tape is initialised at 800 bpi and retained for use in a later job step.

JCL PARAMETERS AND DDNAMES

The tape to be initialised is defined by the symbolic parameter TAPE. If it is to be used in a later job step, the tape may be retained by specifying RETAIN=RETAIN. Options are passed to the program by the symbolic parameter OPT. TLSINIT uses the ddname TLS01 for the tape and TLS03 for messages.

OPTIONS

Options are passed to the program by the OPT symbolic parameter or keyword. They may be separated by '/', ',' or space(s). The default option is 'D1600/E185'. The options available are:-

Dn - density that the tape is to be recorded at. 'n' can be one of: 1600 or 3, 800 or 2.

En - specifies the number of directory entries; 0 < n < 936.

S - scratch - restore the tape to a non-TLS SL tape.

'n' may be separated from the option letter by one or more spaces. If n=0 the option is treated as not being given.

Other illegal options are ignored - and an error given.

NOTES

(a) Once a tape has been initialised for TLS at a certain density, there is no need to specify the density again. TLSINIT ignores any density specified by the DCB parameter on the JCL tape (TLS01 or TAPE) dd statement.

(b) If both the D and S options are given, the D option is ignored.

RETURN CODES

Error messages are sent to TLS03, if present, otherwise a return code of 999 is issued.

0	-	successful run
2	-	unknown or illegal options
8	-	unable to write to the tape or I/O error on Volume Label
32	-	dictionary could not be written.
999	-	insufficient REGION
	-	other serious failures

TLSRECOV

TLSRECOV is a Phoenix command and JCL catalogued procedure which may be used to reconstruct the dictionary of a TLS tape after the loss of the main copy due to errors on the tape. TLSR is a synonym for TLSRECOV (Phoenix only).

PHOENIX EXAMPLES

(a)
```
JOB ABC1 100,TAPE9 ABC101
PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=ABC101
TLSRECOV
```

The tape dictionary is reconstructed using the default options.

(b)
```
JOB SPQR 999,TAPE9 XYZ209,PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=XYZ209
TLSR OPT=DR DSN=XYZ2.DICT
```

This reads the dictionary from the disc file specified by the DSN keyword.

See Example Sheet 21 for more examples.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
OPT	text	NR	-	-
TAPE	ddname	TAPE	-	-
VER	dsd	*	/N	messages dataset
DSN	dsd	&&DSN	/N	see note (b)
STORE	(integer / integer K)	34K	-	-

JCL EXAMPLES

(a)
```
JOB ABC1 100,TAPE9 ABC101,JCL
// EXEC TLSRECOV,TAPE=ABC101
```

This is simple recovery using the default options.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

TLSRECOV 2

(b)
```
JOB SPQR 999,TAPE9 XYZ987,JCL
// EXEC TLSRECOV,TAPE=XYZ987,OPT=D,
//   DSN='XYZ9.FILE',DISP=OLD
```

The dictionary is read from disc.

See Example Sheet 21 for more examples.

JCL PARAMETERS

The tape to be recovered is defined by the symbolic parameter NAME or TAPE. The temporary disc dataset used to hold the reconstructed dictionary is defined by the symbolic parameter DSN, and DISP should be specified to use or create a catalogued dataset. Output is sent to a dataset defined by the symbolic parameter PRINT1. For a full list of the parameters available print the file INFO.JCL.PARMS(TLSRECOV).

OPTIONS

Options are passed to the program by the OPT symbolic parameter, and may be separated by spaces, commas or /. The default value is S4. The options available are:-

D — specifies that the dictionary be read from disc and not reconstructed from the tape labels. The dictionary would have been placed there by a previous run of TLSRECOV.

Fn — where 0 < n < 936
specifies the number of user datasets to be entered in the reconstructed dictionary. This must be specified if the tape is known not to be correctly terminated with a double tapemark.

Sn — where n > 3 (unless option L is specified)
specifies the absolute (BLP) dataset to skip to before scanning for user datasets. This dataset should contain the header labels of the first (SL) user dataset that is to be placed in the reconstruced dictionary. This must be used if the tape has been overwritten by an accidental run of TLSINIT etc.

Ln — where 0 < n < 936
specifies that the dictionary labels are not to be verified and that the reconstructed dictionary is not to be written back to the tape. The value n specifies the size of the dictionary. This option causes the tape to be opened BLP and should be used for serious tape failures or when the dictionary has been corrupted or overwritten.

NOTES

(a) '/' commas or spaces may be used as separators between options; in this case the whole string should be enclosed within quotes.

TLSRECOV
3

(b) The recovery procedure should be used only after the messages 'TLS Dictionary cannot be read' or 'TLS dictionary cannot be written back' are printed by the main TLS program. The 'R' option can normally be used; in most cases, the dictionary will be written back successfully (the program prints a message to confirm this).

It is advisable to allocate a permanent dataset for the disc file by means of the DSN symbolic parameter; if the dictionary is successfully written back to tape, the disc file can then be deleted. If, however, a permanent I/O error prevents the dictionary from being written back, the contents of the tape can be copied onto another TLS tape by TLSCOPY, using the disc resident copy of the dictionary.

(c) If an error occurred on the last TO TAPE transfer, the 'F' parameter must be used to terminate scanning of the tape; the value of 'n' is the number of entries used, as printed under 'Tape usage'.

(d) If any option is specified, the default options N and R no longer apply.

RETURN_CODES

TLSRECOV issues the following return codes:-

0	-	successful run
2	-	unknown options
4	-	entries lost
8	-	I/O error whilst reading tape
	-	invalid disc dictionary format
20	-	dictionary full
32	-	dictionary failed to be written to tape
999	-	tape is not a TLS tape
	-	insufficient REGION
	-	other serious failures

TLSSCAN

TLSSCAN is a Phoenix command and JCL catalogued procedure which scans a TLS tape for versions of a dataset specified by the keyword or symbolic parameter NAME.

If the dataset is partitioned then a memberlist of each version is given, otherwise the first few lines of each version are printed. For each version a heading is printed which gives:

1. The filenumber on the tape.
2. The TLS version number.
3. The tape and disc creation dates.

PHOENIX EXAMPLES

(a)
```
JOB PC99 1234,TAPE9 PC9911
PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=PC9911
TLSSCAN NAME=PC99.DATASET
```

scans PC9911 for all versions of PC99.DATASET.

(b)
```
JOB PC99 1234,TAPE9 PC9912
PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=PC9912
SET PC90
TLSSCAN NAME=.LINKLIB OPT=V3
```

scans PC9912 for the last three versions of PC90.LINKLIB.

The SET name is used to construct the full dataset name.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
NAME[4]	dsn	-	-
OPT	text	-	-
TAPE	ddname	TAPE	-
OUTPUT[5]	dsd	*A	/N/VBA
STORE	(integer / integer K)	68K	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

4 Item may not be omitted.
5 Keyword may not be omitted

JCL EXAMPLES

(a) JOB SPQR 1234,TAPE9 SPQR01,JCL
// EXEC TLSSCAN,TAPE=SPQR01,NAME='PC99.DATASET'

This scans SPQR01 for all versions of PC99.DATASET.

(b) JOB SPQR 1234,TAPE9 SPQR02,JCL
// EXEC TLSSCAN,TAPE=SPQR02,NAME='SPQR.LINKLIB',OPT=V3

This scans SPQR02 for the last three versions of SPQR.LINKLIB.

JCL PARAMETERS

The symbolic parameter TAPE defines the tape volume to be scanned.
The name of the tape dataset is defined by the symbolic parameter NAME.

OPTIONS

Options are passed to the program by the OPT symbolic parameter or keyword. They may be separated by slash (/), comma or space(s). The default value is OPT='L7/V0'. The options available are:

Vn - the number of Versions scanned (i.e. V2 = versions 0 and -1)
Ln - the number of Lines printed per version (0<n<51)

'n' may be separated from the option letter by one or more spaces.
If n is zero, the option is treated as not having been given.

Other illegal options are ignored and an error given.

RETURN CODES

Error messages are sent to SYSPRINT, if present, otherwise a return code of 999 is issued. A return code greater than 4 causes scanning to be terminated. The following are the return codes issued:

0 - successful run.
2 - unknown option(s).
4 - I/O error on a data block.
8 - I/O error on labels, incorrect labels etc..
12 - no files on tape, insufficient REGION size or bad dataset name.
16 - tape cannot be opened.
999 - error but unable to give message.

TO

TO is a Phoenix command which may be used to change control to a command sequence without changing the command level or the values of local variables and parameters. On exit from the command sequence control is returned to the previous level, not to the command following TO.

PHOENIX EXAMPLES

(a)
```
LOCAL X=AB PRINTER=1K
...
TO .COMMANDS
```

Control is transferred to the dataset .COMMANDS and the symbolic parameters X and PRINTER remain set.

(b)
```
IF KX EQ 10 TO .COMM10
TO .COMM9
```

If the variable KX is equal to 10 control is transferred to the dataset .COMM10. Otherwise control is transferred to .COMM9

PHOENIX COMMAND SYNTAX

(a) TO dsd[1] ($\begin{matrix}P\\PARS\end{matrix}$)[1] parameter-declaration

(b) TO dsd[1] (command environment) parameter declaration

NOTES

The P or PARS keyword may be optionally followed by a parameter declaration which is used to set local parameters in the command sequence.

The P or PARS keyword is only needed if there is no dsd or if the first parameter in the parameter declaration is P or PARS.

Choose one of the alternatives in brackets.
1 Optional.

TRANS

TRANS is a Phoenix command and JCL catalogued procedure for translating the characters in a sequential dataset (which may be a member of a PDS) from any character code into any other character code. It can change the blocksize, record length or record format of the dataset, and remove or add trailing spaces as desired. There are additional facilities designed for use with paper tape in non-standard character codes. The program deals with the following record formats:

F, FB, FBS, V, VB, VBS (but not LRECL=X), U.

Carriage control characters may be present, but are not treated specially.

PHOENIX EXAMPLES

(a) TRANS

The current document is translated from BCD to EBCDIC; output is to the new current document.

(b) TRANS .ASCII.FILE TO .EBCDIC.FILE CODE ASCIEBCD

This translates an ASCII file to EBCDIC.

(c) TRANS .AFILE TO .BFILE WITH .CODETBL

This translates .AFILE according to a user supplied translation table.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
FROM	dsd	%C	/R	-
TO	dsd	%O	/N	-
VER	dsd	*	/N	verification
CODE[5]	membername	BCDEBCD	/R	see note below
WITH[5]	dsd	-	/R	-
(OPT / PAD)	text	-	-	-
STORE (integer / integer K)		32K	-	-

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Key may not be omitted.

TRANS
2

PHOENIX NOTES

A set of commonly used translation tables is provided for convenience in a file called SYS2.CODE. A complete list of currently available translation tables may be obtained by running the XPDS program on SYS2.CODE. The keyword CODE may be used to select a member of this file.

eg. CODE BCDEBCD
CODE UPCASE

If the user wishes to supply his own translation table, the keyword WITH may be used to specify the dataset containing it. See notes below.

CODE and WITH are mutually exclusive.

JCL EXAMPLES

(a)
```
// EXEC TRANS
//TO DD SYSOUT=C
//FROM DD *
 card deck
//*
```

translates an instream card deck from BCD to EBCDIC and prints the output.

(b)
```
// EXEC TRANS,CODE='(ICLEBCD)'
//TO DD DSN=PS99.SOURCLIB(ICL),DISP=OLD
//FROM DD DSN=PS99.ICLCARD.DECK,DISP=SHR
```

translates from ICL card code to EBCDIC and stores the output as a member of a partitioned dataset.

(c)
```
// EXEC TRANS,CODE='(FLEX8)',OPT='SH7A7C,00'
//FROM DD DSN=ABC1.FILE,DISP=SHR
//TO DD DSN=ABC1.NEW.FILE,DISP=(,CATLG),
// SPACE=(TRK,(1,1)),UNIT=DISC,DCB=VB
```

translates to EBCDIC a binary disc dataset which has been created from a paper tape punched in a non standard code. Two shifts were used, upper case (hex code 7A) and lower case (hex code 7C). '00' is a fill character (see below) for removing runout.

JCL PARAMETERS AND DDNAMES

The dataset to be translated is described in a dd statement named FROM. The translated output is sent to the dataset defined by the ddname TO. If any of the DCB parameters RECFM, LRECL or BLKSIZE are not specified, they are taken to be the same as the corresponding parameters for the input. DCB parameters should always be specified when creating a disc file from card input.

TRANS
3

The name of the translation table is passed to the program by the symbolic parameter CODE which refers to members of the pds SYS2.CODE, which contains a set of commonly used translation tables

eg CODE='(BCDEBCD) translates from BCD to EBCDIC
'(EBCDBCD)' EBCDIC to BCD
(ICLEBCD) ICL card code to EBCDIC
(EBCDICL) EBCDIC to ICL card code
'(UPCASE) lower case letters to upper case

If the user wishes to supply his own translation table then the symbolic parameter NAME can be used to specify the dataset containing it. See NOTE below.

OPTIONS

These are specified in the OPT symbolic parameter The options are separated by spaces or commas

(a) the 'fill character OPT= xy

xy is a two-digit hexadecimal number The 'fill' character used by TRANS is set to be the character corresponding to hexadecimal xy If the output RECFM is variable or undefined trailing occurrences of the 'fill character are removed from all records. If the output RECFM is fixed, then the fill character is used to pad output records The default fill character is the 65th character in the first translation table (normally blank)

(b) 'shift' characters OPT=SHxyxyxyxy

Each xy is a two-digit hexadecimal number defining a shift character. As many shift characters as required may be supplied One translation table is read from the CODE dataset for each shift character supplied. Before the FROM dataset is read TRANS is set to use the first translation table During input, each record is scanned for shift characters If the i^{th} shift character is encountered, TRANS is set to translate subsequent characters according to the i^{th} translation table. The shift character itself is omitted from the record A shift character remains in force until another shift character is encountered, even across record boundaries.

MESSAGES

No messages are produced when TRANS runs successfully

NOTES

If a user wishes to supply his own translation tables, the following points should be noted.

1. The dataset must consist of one or more fixed length 256 byte records, each containing a translation table.

2. Only one table is required unless the special paper tape oriented facilities are being used.

RETURN_CODE

A return code of 12 is given if a dataset cannot be opened, or an I/O error occurs.

TRANSFER

TRANSFER is a Phoenix command and JCL catalogued procedure which copies one or more datasets between disc and magnetic tape. Partitioned datasets are copied in the format used by the IBM utility program IEHMOVE.

PHOENIX EXAMPLES

(a)
```
JOB PC99 1234,TAPE9 PC9900
PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=PC9900,LABEL=(,SL),DISP=OLD
TRANSFER OPT 'FROM TAPE 3 .PDS AS &PDS'
```

Recovers a pds from file 3 of the tape, copying it to a temporary dataset.

(b)
```
JOB PC99 1234 UNLOAD PDS,TAPE9 02PC99
PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=02PC99,LABEL=(,NL),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),DISP=OLD
TRANSFER WITH %H& ENV FRED=.SEQ/N/VB/PS
FT 1 .PDS
FT 2 .SEQ
&
```

Recovers a pds and a sequential dataset from an unlabelled tape. Commands are given in an instream dataset. The file .SEQ is pre-allocated to give it the correct DCB parameters.

(c)
```
JOB PC99 1234,TAPE9 05PC99
PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=05PC99,DISP=OLD,LABEL=(,NL)
TRANSFER OPT 'TT 1 .PDS;TT 2 .SEQ'
```

This job writes a pds and a sequential dataset to an unlabelled tape.

(d)
```
JOB SPQR 1234,TAPE9 SPQR01
PHX
//TAPE DD UNIT=TAPE9,VOL=SER=SPQR01,DISP=OLD
TRANSFER WITH %H% ENV DD1=ZZZZ.TEST/N/VB/PS
NOSTOP;SET ZZZZ
FT 1 SPQR.RES AS .R1
FT 2 SPQR.TEST AS DD1
TT 10 .PROG AS SPQR.PROG
%
```

This job recovers two sequential files from a labelled tape and stores them on ZZZZ's filespace; another file is transferred to tape.

TRANSFER
2

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
(SYSIN / WITH)	dsd	-	/R
TAPE	ddname	TAPE	-
VER [5]	dsd	*	/N
OPT	text	-	-
STORE	(integer / integer K)	50	-
ENV[5]	text	-	/R

JCL EXAMPLES

(a) JOB PC99 999,TAPE9 PC9900,JCL
// EXEC TRANSFER,TAPE=PC9900,OPT='FROM TAPE 3 .PDS AS XYZ'
//XYZ DD DSN=&PDS,UNIT=DISC,DISP=(,PASS),SPACE=(TRK,(5,5,2))

Recovers a pds from file 3 of the tape, copying it to a temporary dataset.

(b) JOB PC99 999,TAPE7 01PC99,JCL
// EXEC TRANSFER,TAPE=01PC99,DEN=1,TRTCH=ET,
// UNIT=TAPE7,LABEL=NL
//FRED DD DSN=PC99.SEQ,UNIT=DISC,DISP=(,CATLG),DCB=VB,
// SPACE=(TRK,(2,2),RLSE)
//SYSIN DD *
FT 1 .PDS
FT 2 .SEQ
//*

Recovers a pds and a sequential dataset from an unlabelled tape. Commands are given in an instream dataset. The file .SEQ is allocated via a dd statement to give it the correct DCB parameters.

(c) JOB PC99 1234,TAPE9 05PC99
JCL
// EXEC TRANSFER,TAPE=05PC99,OPT='TT 1 .PDS;TT 2 .SEQ',LABEL=NL

This job writes a pds and a sequential dataset to an unlabelled tape.

- -
Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.
- -

TRANSFER
3

JCL PARAMETERS

The symbolic parameter TAPE defines the tape volume to be used. All other parameters are optional.

LABEL=SL or NL	UNIT=TAPE9 or TAPE7
DEN=omitted, 1, 2, 3	TRTCH=omitted, C, E, T, ET

Default values are SL, TAPE9.

COMMANDS

The program takes commands from the OPT field, if present, and then from the dataset defined with ddname SYSIN (JCL) or by keyword SYSIN or WITH (Phoenix). Commands are delimited by semi-colon or end of record. DCB parameters on the TAPE dd statement are ignored if the tape is SL; otherwise they provide default values to be used as described below.

FILESIZE n FILESIZE n,m	This command is used to set up default space parameters to be used if a tape to disc copy is requested for a sequential dataset which does not pre-exist and for which there is no dd statement. n is the primary allocation in tracks, and m is the secondary allocation (default 0). The default setting is 2,2. Note that for a pds, the space parameters are determined from information written on the tape file.
FT n dsn FT n dsn AS B	This command is used to copy a file from tape to disc. 'n' is the tape file sequence number. 'dsn' is the name of the tape dataset and the disc dataset to which the transfer is made (unless an AS clause is used); for SL tapes it must match that written on the tape. B may be a dsname or ddname which refers to the disc file which is to be written to. A dd statement is mandatory only for OS temporary files or Phoenix anonymous files (see notes). FT is an abbreviation for FROM TAPE. For unlabelled tapes a dummy dsn must be given if an AS clause is specified.
NOSTOP	This command is used to tell the program to continue execution even if a previous command has ended in failure. Initially NOSTOP is not set.
NOVERIFY NV	This command cancels a VERIFY command. The initial setting is NOVERIFY. NV is an abbreviation for NOVERIFY.

TRANSFER
4

SET string SET NAME string	This command is used to set up a string of up to 17 characters for use as a prefix to dataset names which begin with a dot ('.'). 'NAME' is optional. The string is initialized to the Phoenix 'set' string, or the userid, as appropriate.
STOP	This command causes TRANSFER to terminate execution if an error occurs while interpreting a command. By default STOP is set.
TT n dsn1 TT n ddname TT n dsn1 AS dsn2 TT n ddname AS dsn2	This command is used to copy a file from disc to tape. 'n' is the tape file sequence number; dsn1 and ddname identify the disc file to be copied. A dd statement is mandatory only for JCL temporary or Phoenix anonymous files. The AS clause, which is optional, defines a new name for the tape dataset; if it is omitted the tape dataset is given the same name as the disc dataset. An AS clause is recommended when the transfer is of a Phoenix or JCL temporary dataset to a labelled tape.
VERIFY V	This command causes TRANSFER to reflect subsequent commands and to print a message after each file transfer to confirm that the command has been obeyed successfully. The initial setting is NOVERIFY. V is an abbreviation for VERIFY.

NOTES

Action is taken by the TRANSFER program as follows:-

(a) DD statement provided for an FT command

If the dataset pre-exists, it must have been given a dataset organisation. For a new dataset, DSORG may be set explicitly in the DCB or implicitly by the presence or otherwise of a request for a number of pds directory blocks. If DSORG=PS a sequential copy is performed. For SL tapes, DCB parameters for the disc file must either be unset or be compatible with those of the tape file (for example, if the tape dataset has RECFM=U then the disc dataset must have RECFM=U and a blocksize at least as large as the tape dataset). For non-SL tapes, DCB parameters are taken from the disc file if those are set; otherwise the tape file is treated as DCB=(RECFM=U,BLKSIZE=maximum possible for the disc). If the disc file has DSORG=PO, the tape file is assumed to consist of an unloaded pds in 'IEHMOVE' format. An error message results if this is not the case. If DCB parameters exist for the disc file, they must be compatible with those written on the 'IEHMOVE' file.

TRANSFER
5

(b) No DD statement provided for an FT command

If the disc file pre-exists, and has a DSORG, a dynamic dd is set up for it (DISP=OLD) and the program proceeds as for (a) above. If the tape is SL, or non-SL with DCB parameters given on the dd statement as DCB=(RECFM=FB,LRECL=80,BLKSIZE=800), a check is made to see whether the tape file contains an unloaded pds. If so, the disc file is created dynamically with SPACE and DCB parameters obtained from information in the 'IEHMOVE' file. If not, the file is created with SPACE=(TRK,(2,2),RLSE) and the DCB parameters of the tape file (default is DCB=(RECFM=U,BLKSIZE=13030)).

TRUE

TRUE is a Phoenix flag which may be tested by the user. It is always set to true. This flag is useful when used in conjunction with the IF statement.

EXAMPLE

```
IF <P1> SKIP ON
* P1 is false
QUIT 100
*ON
.....
QUIT
```

If P1 has the value TRUE control is transferred to the label ON and the command sequence continues to be obeyed. Otherwise control is transferred to the calling level of command.

TYPE

TYPE is a Phoenix command which types a sequential dataset or a member of a partitioned dataset without line numbers.

PHOENIX EXAMPLES

(a) TYPE

The current document is typed.

(b) TYPE .A OPT=V

This types .A with verification of the DCB characteristics.

(c) TYPE .B:C

A member of a partitioned dataset is typed.

(d) TYPE &A

This types a temporary dataset.

(e) TYPE %M

This types a system dataset (the messages document).

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Notes
FROM	dsd	%C	/R	-
OPT	text	-	-	options
STORE	(integer / integer K)	32K	-	-

OPTIONS

The options available are the same as those for FILE.

NOTE

TYPE sends its output to the * dataset.

Choose one of the alternatives in brackets.
Any item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

UNLESS

UNLESS is a Phoenix control command which is used to make tests in a command sequence. IF NOT and IF - are synonyms for UNLESS.

PHOENIX EXAMPLE

```
        .
        .
    UNLESS ONLINE SKIP A
    ABANDON
    *A
        .
        .
```

If the ONLINE switch is set, the command sequence is abandoned and command returns to the console, otherwise control is transferred to the label A.

PHOENIX COMMAND SYNTAX

```
    UNLESS condition
```

See IF and Section 4.10.5 for further details.

USER SWITCHES

Sixteen switches which may be set and tested by the user are available globally. The switches are numbered 0 to 15 inclusive.

PHOENIX EXAMPLES

(a) SW 2

sets user switch 2

(b) SW -4

unsets user switch 4

(c) SET -SW 6 or SET SW -6

unsets user switch 6

(d) SET SW 11

sets switch 11

(e) IF SW1 JUMP LABEL

If switch 1 is set control is transferred to LABEL

(f) UNLESS SW9 SET SW4

If switch 9 is not set, switch 4 is set.

PHOENIX COMMAND SYNTAX

```
        SET  SW  n
  or
        SET  SW  -n
  or
        SET  -SW  n
  or
        SET  -SW  -n
  or
        SW   n
  or
        SW   -n
```

SW may be replaced by SWITCH.

VARIABLES

There are ten global arithmetic variables, %G0 to %G9, and ten local arithmetic variables, %L0 to %L9. When the global variables are set, they stay set until the end of the job or console session. Local variables remain set only until the end of the current command level. Variables may be set to full word integers, text or other variables. For further details of local and global parameters see Section 4.9.4.

PHOENIX EXAMPLES

(a) SET %G3=10

sets the global variable %G3 to 10.

(b) PARS %G1=%G4-%L3

sets %G1 to the difference between the variables %G4 and %L3.

(c) SET %L1='text'

SET P %L1=text

These are equivalent ways of setting %L1 to the value text.

(d)
```
SET %G1=5
*BACK
...
PARS %G1=%G1-1
IF %G1 GT 0 JUMP BACK
```

Here a loop to be obeyed 5 times is constructed using the variable %G1.

PHOENIX COMMAND SYNTAX

Global variables

SET %Gn=item[±item] or PARS %Gn=item[±item]

Local variables

SET %Ln=item (±item) or PARS %Ln=item(±item)

An item is an integer, a global or local variable or a quoted text.

VIEW

VIEW is a Phoenix command which may be used at special Phoenix consoles to inspect graphical output stored on a disc file which has been generated by the GRAPHICS.FORTLIB subroutine package. Commands are supplied to the program interactively. Terminals which can be used for this purpose include the Tektronix models 4006, 4010 and 4012 display terminals, or any programmable terminal running a suitable emulator, such as that for the DEC GT40.

PHOENIX EXAMPLES

(a) VIEW .PICTURE

The file .PICTURE is drawn on the screen; directives may be typed in by the user.

(b) VIEW FROM .GRAPH OPT IILDG

The file .GRAPH is drawn on the screen subject to the directives specified in the OPT field. Further directives may be typed in.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Notes
FROM	dsd	%C	Graphical input
WITH	dsd	*	Control input
TO	dsd	*	Graphical output
VER	dsd	*	Control output
OPT	text	-	Initial Control input

COMMANDS

Directives are read first from the 'OPT' field (if it exists) and then from the 'WITH' stream which is normally from the terminal. A picture (as defined in note (a)) is drawn after any output command (?, E or G) or at the end of the line in which it was selected (by *, +, N or M). Thus window setting and pen selection commands can be effected before the picture is drawn.

+ The next picture is selected.

* The input is restarted from the first picture.

Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.

?	The current picture is redrawn through the current window.
A	Rotate the new window 90 degrees Anticlockwise.
B	Draw the Boundary of the defined drawing area.
C	Rotate the new window 90 degrees Clockwise.
D Dx	Move the new window Down. The default movement is half the window height, but a fraction of the height may be given as the real argument x.
E	Draw the Entire picture at the largest reasonable scale.
F	Abandon (Forget) the new window, resetting it to the current window.
G	Go. The picture is redrawn as seen through the new window, which becomes current.
H	Return the X axis to Horizontal, with the Y axis vertical.
I Ix	Move the new window In. The default movement is by half the window size, but a ratio in which the window is to be reduced may be given as a real argument, x.
L Lx	Move the new window Left. The default movement is by half the window width, but a fraction of the width may be given as a real argument, x.
Mn	Select the picture number n. The directive is followed by an integer picture number, and that picture is next selected, if this can be done without rewinding the stream.
N	The Next picture is selected.
O Ox	Move the new window Out. The default movement is to twice the window size, but a number by which the window size is to be multiplied may be given as a real argument.
P	Draw only those lines of specified colour(s). The colours available are B or 1 for Black, R or 2 for Red and G or 3 for Green. The default is BRG. The list of colours is terminated by end of line.
Q	Quit. The command is terminated.
R Rx	Move the new window Right. The default movement is by half the window width, but a fraction of the width may be given as a real argument.
U	Move the current window Up. The default movement is by half the window heightbut a fraction of the height may be given as the argument x.

VIEW 3

V Information relevant to the current picture is typed. This includes the number, the stream title, the limits of the plot and details of the windows.

W The screen is cleared and the command terminated.

Xn The X directive takes an integer parameter which specifies the length in device units below which consecutive line segments will be merged to reduce the drawing time. The default value is 1.

NOTES

(a) A picture is defined as a piece of graphical material delimited by 'create stream' (CRTPLT) and 'break stream' (BRKPLT) requests, or by the end of the document.

(b) The current window is defined as the window used for drawing the current picture. It is initially set so that the screen width is filled by a picture which would fit onto the narrow plotter.

(c) The new window is the window which will be used for drawing the next view of the current picture.

WAIT

WAIT is a Phoenix control command which, if obeyed in a console session, causes a pause of an integer number of seconds before the next command is obeyed and which is ignored in a job.

PHOENIX COMMAND SYNTAX

WAIT n[1]

n is an integer.

EXAMPLE

```
*BACK
WAIT
STATUS TO $
UNLESS RC JUMP BACK
SUBMIT
```

The loop is repeated with a ten second wait until STATUS gives a non-zero condition code, when the SUBMIT command is obeyed.

NOTE

If n is less than 10 in the syntax above, a message is printed, and n is set to 10. If n is greater than 300, n is set to 300, but no message is output.

---

1 Optional

XLABEL

XLABEL is a Phoenix command and JCL catalogued procedure which prints information contained in the label of a disc dataset. This label, which is also known as the dataset control block (DSCB) contains information about the dataset such as the record format, blocksize and logical record length. All fields that can be set by LABEL are formatted. An extent list is output if the X option is set. XLABEL is mainly useful for investigating errors.

PHOENIX EXAMPLES

(a) XLABEL .FILE

Information for .FILE is printed.

(b) XLABEL NAME=.PROG TO=&A OPT X

Information for .PROG, including an extent list is filed in the temporary dataset &A.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
NAME	dsd	-	/R
TO	dsd	*	/N
OPT	-	-	-
STORE	(integer / integer K)	14	-

JCL EXAMPLES

(a) // EXEC XLABEL,NAME='SPQR.FILE'

Information for the dataset SPQR FILE is printed.

JCL PARAMETERS AND DDNAMES

The input dataset is defined by the symbolic parameter name. Verification output is sent to a dataset defined by ddname SYSPRINT. Options are passed to the program by the symbolic parameter OPT.

XLABEL
2

RETURN_CODES

The program issues the following return codes:

0	successful run.
4	invalid options.
8	dataset not a disc file or does not exist.
12	system error.
16	verification dataset is absent.

XMODULE

XMODULE is a Phoenix command and JCL catalogued procedure which displays pds directory entries. The overlay tree for suitable modules may be displayed on request.

PHOENIX EXAMPLES

(a) XMODULE NAME=.SAVE OPT=MEMA,MEM

The directory entries for .SAVE:MEMA and .SAVE:MEM are printed.

(b) XMODULE NAME=.PDS WITH=&B TO=&C OPT=.O

The directory entries for the members listed in the temporary dataset &B are output to &C together with an overlay tree for each member.

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
NAME	dsd	-	/R
WITH	dsd	%N	/R
TO	dsd	*	/N
OPT	text	-	-
STORE	integer integer K	20	-

JCL EXAMPLES

(a) // EXEC XMODULE,NAME='SPQR.SAVE',OPT='MEMA,MEM'

The directory entries for SPQR.SAVE(MEMA) and SPQR.SAVE(MEM) are printed.

(b) // EXEC XMODULE,NAME='SPQR.PDS',WITH=&B,OPT=O

The directory entries for the members listed in the temporary dataset &B are printed, together with an overlay tree for each member.

JCL PARAMETERS AND DDNAMES

The input pds, is defined by the symbolic parameter NAME. A member list and options may be provided by either the symbolic parameter OPT or WITH (see note (a) below). Output from the program is defined by a dd statement with ddname SYSPRINT.

XMODULE
2

OPTIONS

The default option is LOAD; the following options are provided:

OVLY or O	print the overlay tree if any.
LOAD or L	treat load modules specially, but do not print the overlay tree (default).
TEXT or T	treat all entries as text and print in hex.
QUIT or Q	stop processing.

NOTES

(a) Input is taken firstly from the PARM field (OPT symbolic parameter) and then from the SYSIN dataset (WITH symbolic parameter).

(b) Membernames may be separated by spaces, commas, semicolons, slashes or newlines. If these are in quotes, they will not be checked for possible options and may contain any character.

XPDS

The Phoenix command or JCL catalogued procedure XPDS has the following functions:

(a) to provide a list of members of a pds, together with their aliases, if any.

(b) to alter the directory of a pds by deleting entries, renaming members, or creating aliases.

(c) to print the number of directory blocks used and left.

PHOENIX EXAMPLES

(a) XPDS PS99.PDS

types a list of members of the pds in the default output format.

(b) XPDS PS99.PDS OPT='D PROG'

deletes member PROG from the directory, then types a member-list as above.

(c)
```
XPDS KE97.LIB INPUT *
ALIAS PROG1 AS PROGA;DELETE PROGB,PROGC
PREFER PROGD AS MAIN
W
```

This is interactive use of XPDS. A list of members follows after the W is typed.

PHOENIX NOTE

Any control statements given following OPT or DATA are obeyed before the control statements contained in the datasets following INPUT or WITH.

XPDS
2

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier	Note
NAME	dsn	-	/R	mandatory
(INPUT WITH)	dsd	$	/R	-
TO	dsd	*	/N	-
(OPT DATA)	text	-	-	-
STORE	(integer integer K)	40K	-	-

JCL EXAMPLES

(a) // EXEC XPDS,NAME='SPQR.LIB'

types a list of the members of SPQR.LIB in the default output format

(b) // EXEC XPDS,NAME='SPQR.LIB',OPT='D PROG'

deletes member PROG from the directory, then types a member-list as above

(c)
```
// EXEC XPDS,NAME='SPQR.LIB'
//SYSIN DD *
 ALIAS PROGA AS MAINPROG; DELETE PROGC
//*
```

commands are obeyed, then a list of members is typed.

JCL PARAMETERS AND DDNAMES

The name of the pds to be examined must be given as the value of the symbolic parameter NAME while the options or control statements may be given either as a value of the symbolic parameter OPT or in the dataset defined by ddname SYSIN. If neither is present, the directory is not altered and the output consists only of a list of members.

For a full list of the parameters available print the file INFO.JCL PARMS(XPDS)

Any control statements given by the OPT symbolic parameter are obeyed before the control statements introduced by the ddname SYSIN.

Any key or item may be omitted unless otherwise stated
Choose one of the alternatives in brackets.
Keyed items may occur in any order; unkeyed items in the order given

---

XPDS
3

OUTPUT FORMATS

There are four formats:

(a) default format: alias entries follow their main name (across the page) preceded by '=' two columns before the start of the name. A blank entry is left at the start of a line if the first entry on that line is an alias, and at the end of a line if the next entry is a main name which has at least one alias.

(b) alphabetical format as specified by ALPHABETIC below.

(c) one-column format as specified by ONECOLUMN below.

(d) alphabetic one-column format; all entries in alphabetic order in a single column, with no heading or directory information. Aliases are marked by '*' in column 9 and their main name given in columns 11-18. Specify AB;OC.

RETURN CODES

The following return codes are issued:

0 no errors

8 syntax errors
member not found (but not for delete)
member already exists
directory full

12 I/O error
XPDS error (should not occur)
dataset not a pds
not enough store
corrupt directory (unexpected end of file)

16 open failure.

CONTROL STATEMENTS

The following control statements are provided. Items in brackets are optional, and can be omitted. If no control statements are supplied, a list of members and their aliases are output in the default format.

ALIAS	name1 (AS) name2
ALPHABETIC	
BACKWARDS	
DELETE	name1
DELETEALIAS	name
DELETEMULTIPLE	n-m name
DELETEMULTIPLEALIAS	n-m name
DSIZE	

XPDS
4

```
END
LIST
MAINNAME        alias (AS) main name
NOLIST
NOPROMPT
ONECOLUMN
PREFER          name1 (AS)  name2
PREFERALIAS     member1 (AS)  member2
PROMPT
RENAME          name1 (AS)  name2
RENAMEMULTIPLE  a-b n-m name
SORTTTR
SSI  (member name (data))
STOP
TTR
```

ALIAS or A

This creates a new entry in the directory, name2, as an alias of name1. It fails if name2 already exists or name1 does not exist.

ALPHABETIC or AB

All entries are printed in alphabetic order, aliases followed by '*' eight columns after the start of the name. See also OUTPUT FORMATS note (d) on the previous page.

The options ALPHABETIC and SORTTTR are mutually exclusive; if both are given, the later one will be used.

BACKWARDS or B

The member list is output in reverse order. However, unless ALPHABETIC is specified, aliases still follow their main name in alphabetic order.

DELETE or D

This removes the entry name1 from the directory. It may be an alias or a 'real' entry. Note that deleting a 'real' entry does not delete its aliases as well. This command does not fail if the member or alias does not exist, but a message is printed.

DELETEALIAS or DA

This deletes a member and all its aliases; more precisely, it deletes the given name and all names referring to the same member from the pds directory.

DELETEMULTIPLE or DM

This always takes two arguments; the first is of the form n-m or n:m:.... (to a maximum of four) where n and m are hexadecimal digits, and the second argument is an eight character name, e.g.

```
DM 0-2 IABCDE02  deletes IABCDE00,IABCDE01,IABCDE02
as does  DM 0:1:2 IABCDE02
```

XPDS
5

DELETEMULTIPLEALIAS or DMA

This is similar to DELETEMULTIPLE above, but all aliases of each name are also deleted.

DSIZE or DZ

Type the directory size, then read the next directive.

END or W

This has the same effect as reaching the end of SYSIN, which is to finish the command after listing the member names and giving the directory size.

LIST or L

Type the list of members and the directory size, then read the next directive.

MAINNAME or MN

This enables an alias without a main member name (indicated by ********) to be given a new main name.

NOLIST or NL

This suppresses the listing of members and aliases normally produced after all commands have been processed.

NOPROMPT OR NOPR

Switch off the prompt character > (this is the default).

ONECOLUMN or OC

This causes output in one column only, with no heading or directory information. Aliases follow their main name and are marked by '*' in column 9. This format is intended for post processing by other programs. If there are no entries in the directory, the output consists of one blank line only. See also OUTPUT FORMATS note (d) on page 2 of XPDS.

PREFER or P

This has the same effect as ALIAS, but in addition it performs an implicit delete if the second name already exists.

PREFERALIAS or PA

This moves all the aliases from one member to another. Either name given on the command may be the main member name or any one of its aliases.

PROMPT OR PR

Switch on the prompt character (>).

RENAME or R

The member or alias name1 is renamed as name2. This fails if name2 already exists or name1 does not exist.

RENAMEMULTIPLE or RM

This provides a concise means of renaming modules by changing the initial letter only. There is a distinction between the first set of arguments, and succeeding ones. Any set may contain three arguments, in which case the syntax is

```
RM a-b  n-m  name
```

where a and b are letters, n and m are hexadecimal digits (n:m:... can also be used as for DM).

XPDS
6

If only two arguments are present, then, in the first set of arguments they are assumed to be 'a-b name' (i.e. only one module renamed), but in subsequent argument sets they are assumed to be 'n-m name', the 'a-b' argument carrying over from the previous argument set. In all but the first argument set, optionally only one argument may appear, in which case the 'a-b' argument is carried over from the previous argument set, and only one module is renamed. e.g.

RM I-J 1-2 IABCDE01, IADZFB02, 6-7 IMNK0406

renames IABCDE01 and IABCDE02 as JABCDE01 and JABCDE02 respectively, IADZFB02 as JADZFB02 and IMNK0406 and IMNK0407 as JMNK0406 and JMNK0407.

SORTTTR or STTR

The member list is sorted by TTR, (instead of alphabetically) before being output. This is the order in which the members were written into the pds. Note that this does not cause the TTR's to be printed.

SSI

This causes the system status information for each member (if any) to be given in the listing. If a membername follows the command the System Status Information for the member is output. If both the membername and data item are specified the System Status Information is set for that member. The data must consist of eight hexadecimal digits. The SSI facilities assume that the members contain either source text or standard format load modules.

STOP or Q

This command terminates processing.

TTR

This causes the relative track and record numbers of each member to be given in the listing.

NOTES

(a) Command records are reflected on the printed output unless the input is from a console. If a command is faulted for syntax reasons, the rest of the command line is ignored.

(b) A warning message is given if a name longer than eight characters is encountered; the name is truncated and the execution continues.

(c) Member names containing non-alphanumeric characters must be in quotes.

XPDS
7

(d) Aliases and entry points

There are two programs which can be used to create alias names for load modules - the linkage editor and XPDS. The way in which the alias is created is fundamentally different in each case, and users should take care to ensure that what actually happens is what they intended.

The directory of a partitioned dataset contains an entry for each membername and each alias name. If the dataset is a load module library, the directory entry also contains the relative address of the entry point associated with the name. Thus the membername has two separate functions:-

To identify the member of the pds.

To give the relative entry point of a load module.

The first item of information is used by all programs which refer to the partitioned dataset. The second item is only used when the load module is executed directly, either by calling it from JCL (// EXEC PGM=...), or as a Phoenix command or by using the LOAD, LINK or XCTL macros in an Assembler program (or their equivalent in a high-level language).

If the module is link edited, information about entry addresses is obtained from the external symbol dictionary in the module itself, and not from the PDS directory.

When an alias is created using the linkage editor, it determines whether the module contains an external symbol with the same name. If so, the entry point associated with the alias is the relative address of that external symbol. XPDS, however, operates only on the directory of a partitioned dataset, and when it creates an alias, it copies the entry point data from the original name. Thus, if the module is subsequently link edited, the correct entry point will be found, but if the module is executed under the alias name, the entry point used will be that associated with the original name.

Summary:-

An alias created by XPDS is only another name for the same member of a pds.

An alias created by the linkage editor can in addition have a different entry point associated with it.

The entry point information in the PDS directory is only used when the module is loaded directly into core - it is not used if the module is re-link-edited.

XTAPE

XTAPE is a Phoenix command and JCL catalogued procedure which scans any labelled or unlabelled tape for information about block organisation and sizes and for any errors on the tape.

PHOENIX EXAMPLES

a)
```
JOB PC99 1234,TAPE9 PC9900
PHOENIX
//TAPE DD UNIT=TAPE9,VOL=SER=PC9900
XTAPE
```

scans the standard labelled tape PC9900.

b)
```
JOB PC99 1234,TAPE7 01PC99
PHOENIX
//TAPE DD UNIT=TAPE7,VOL=SER=01PC99,LABEL=(,NL)
XTAPE OPT=X/B
```

scans the 7 track unlabelled tape 01PC99, in BLP mode. The 'X' option is given because the density and parity information is not known

PHOENIX COMMAND SYNTAX

Key	Valuetype	Default	Default Qualifier
OPT	text	-	-
TAPE	ddname	TAPE	-
OUTPUT[5]	dsd	*	/N
STORE	(integer / integer K)	16K	-

JCL EXAMPLES

a)
```
JOB PC99 1234,TAPE9 PC9900,JCL
// EXEC XTAPE,TAPE=PC9900
```

scans the standard labelled tape PC9900

b)
```
JOB PC99 1234,TAPE7 01PC99,JCL
//  EXEC XTAPE,TAPE=01PC99,OPT='X/B',
//  UNIT=TAPE7,LABEL=NL
```

scans the 7 track unlabelled tape 01PC99 as above.

Choose one of the alternatives in brackets.
Any key or item may be omitted unless otherwise stated.
Keyed items may occur in any order; unkeyed items in the order given.
5 Keyword may not be omitted.

XTAPE
2

JCL PARAMETERS

The symbolic parameter TAPE defines the tape volume to be scanned. All other parameters are optional.

LABEL=SL or NL UNIT=TAPE9 or TAPE7

DEN=omitted, 1, 2, 3 TRTCH=omitted, C, E, T, ET

Default values are SL, TAPE9.

OPTIONS

Options are passed to the program by the OPT symbolic parameter or keyword. They may be separated by '/'. The default value is OPT='E1/F32767'. The other options available are:

Fn	-	stop after n files
X	-	for 7 track tapes try all density/parity combinations
B	-	treat as BLP

The following options apply to SL tapes only:

W	-	continue even if wrong volume mounted.
A	-	read each file one block at a time. (NL tapes are always read this way)
V	-	verify the volume label, then proceed as if the tape were NL

The following option applies to NL tapes, tapes processed BLP, or SL tapes processed with option 'V':

E	-	stop after n double tape marks

Other illegal options are ignored - and a message given.

NOTES

1. n consecutive tapemarks are treated as n-1 double tapemarks.

2. 7 track tapes are not easily scanned by any program owing to the various ways in which the hardware can treat characters. Consequently trouble (usually in the form of I/O errors) may be encountered when scanning 7 track tapes.

YES

YES is a Phoenix control command which is synonymous with QUIT 0 i.e. provides a return code of 0. It is useful in connection with interactive command sequences.

EXAMPLE

```
      .
      .
** IS TREE OAK?  ANSWER YES OR NO.
C *
IF %RC EQ 0 SKIP OAK
* NOT OAK TREE
      .
      .
      .
QUIT 999
*OAK
QUIT 5
      .
      .
      .
```

If the reply is YES control is passed to the calling level of command with a return code of 5 set; otherwise the command stream continues to be obeyed.

INDEX

+	see concatenations		
	4.9.2	Ch.4	22
++++	3.4.4	Ch.3	13
+++B	3.4.4	Ch.3	14
+++E	3.4.4	Ch.3	14
+++F	3.4.4	Ch.3	14
+++K	3.4.4	Ch.3	14
+++L	3.4.4	Ch.3	14
+++P	3.4.4	Ch.3	14
+++S	3.4.4	Ch.3	14
+++T	3.4.4	Ch.3	13
+++Z	3.4.4	Ch.3	13
&	see ampersands		
&COMM	4.3.2	Ch.4	8
!	2.2.5	Ch.2	7
!!	2.2.5	Ch.2	7
$	4.3.2	Ch.4	8
* (console output)	4.3.2	Ch.4	8
(comment qualifier)	4.3.3	Ch.4	11
(job card)	2.4.1	Ch.2	12
	see also PHOENIX		
*$	4.3.2	Ch.4	8
***-BREAK	4.11.1	Ch.4	35
	4.11.1	Ch.4	36
	see also break		
***ABANDONED	2.2.4	Ch.2	5
***ERROR	2.2.8	Ch.2	8
***FROM	Part II	123.1	
***MESSAGE FOR YOU	Part II	117	
***NEW MESSAGE OF THE DAY	2.2.6	Ch.2	7
***SESSION INTERRUPTION	4.11.1	Ch.4	36
*A	4.3.2	Ch.4	9
*B	4.3.2	Ch.4	9
*C	4.3.2	Ch.4	9
*P	4.3.2	Ch.4	9
*Q	4.3.2	Ch.4	9
*R	4.3.2	Ch.4	9
*5	4.3.2	Ch.4	9
*7	4.3.2	Ch.4	9
*8	4.3.2	Ch.4	9
-	Part II	26	
	Part II	127	
	Part II	167	
/*	2.3.4	Ch.2	11
/*DEL	2.3.4	Ch.2	11
/*EOF	2.3.4	Ch.2	11
/*JOB	2.3.4	Ch.2	11
/R	3.2.10	Ch.3	10
/W	3.2.10	Ch.3	10
%	Part II	0	
%%	see %C		
%%C	see %C		
%%SPOOL	see %SPOOL		
%A	4.3.2	Ch.4	8

	4.5.4	Ch.4	13
%ABEND	4.9.5	Ch.4	27
	Part II	0	
%C	4.3.2	Ch.4	7
	4.5.6	Ch.4	13
%CARDS	4.3.2	Ch.4	8
%CCPU	Part II	0	
%CDISCIO	Part II	0	
%CHARGE	Part II	0	
%CHECKED	Part II	0	
%CL	4.3.2	Ch.4	8
	Part II	103	
%COMMAND	4.3.2	Ch.4	7
%CPU	Part II	0	
%CRU	Part II	0	
%CTAPEIO	Part II	0	
%D	4.2.3	Ch.4	4
%DATE	Part II	0	
%DAY	Part II	0	
%DAYNAME	Part II	0	
%ERRJUMP	Part II	69	
%DEF	4.2.4	Ch.4	5
%DUMMY	4.3.2	Ch.4	8
%FINISH	4.10.3	Ch.4	29
%Gn	4.9.5	Ch.4	26
	Part II	198	
%H	4.3.2	Ch.4	8
	4.5.8	Ch.4	15
%HOUR	Part II	0	
%IBMDATE	Part II	0	
%JNUMBER	Part II	0	
%L	see %SL		
%LEVEL	4.8.3	Ch.4	21
	4.9.5	Ch.4	27
	Part II	0	
%LIB	4.3.2	Ch.4	7
%Ln	4.9.5	Ch.4	26
	Part II	198	
%M	4.3.2	Ch.4	7
	4.5.6	Ch.4	13
%MIN	Part II	0	
%MONTH	Part II	0	
%N	4.3.2	Ch.4	8
%name	4.7	Ch.4	19
%O	4.3.2	Ch.4	7
	4.5.6	Ch.4	13
%OUTn	4.3.2	Ch.4	8
%P	4.3.2	Ch.4	7
	4.5.6	Ch.4	13
%PARM	4.2.3	Ch.4	4
%PLOT	4.3.2	Ch.4	8
%PRINTA	4.3.2	Ch.4	8
%PRINTC	4.3.2	Ch.4	8
%PRIORITY	Part II	0	
%PROJECT	Part II	0	
%PTAPE5	4.3.2	Ch.4	8
%PTAPE7	4.3.2	Ch.4	8
%PTAPE8	4.3.2	Ch.4	8

%PWARNLEVEL	Part II	0.1	
%QLEVEL	4.9.5	Ch.4	27
	4.11.4	Ch.4	37
	Part II	0.1	
%R	4.3.2	Ch.4	7
	4.5.6	Ch.4	13
%RC	4.9.5	Ch.4	27
	Part II	0.1	
%RCLEVEL	4.9.5	Ch.4	27
	Part II	0.1	
%RDYTEXT	Part II	0.1	
%REGION	4.2.1	Ch.4	4
	Part II	0.1	
%RESOURCE	Part II	0.1	
%S	4.2.1	Ch.4	4
%SCC	4.9.5	Ch.4	27
	Part II	0.1	
%SEC	Part II	0.1	
%SET	4.9.5	Ch.4	27
	4.11.4	Ch.4	37
	Part II	0.1	
%SL	4.3.2	Ch.4	8
	Part II	145	
%SPOOL	4.10.3	Ch.4	30
%T	4.2.2	Ch.4	4
%TAPEIO	Part II	0.1	
%TERM	4.3.2	Ch.4	8
%TERMINAL	4.3.2	Ch.4	8
%TIME	4.2.2	Ch.4	4
	Part II	0.1	
%UCC	4.9.5	Ch.4	27
	Part II	0.1	
%USER	4.9.5	Ch.4	27
	4.11.4	Ch.4	37
	Part II	0.1	
%WIDEPLOT	4.3.2	Ch.4	8
%WIDTH	Part II	0.1	
%XL	4.3.2	Ch.4	8
%YEAR	Part II	0.2	
?	App D	5	
	App D	6	
	App D	9	
: (line concatenation)	4.7	Ch.4	19
?	2.2.5	Ch.2	6
	App B		
	App D	3	
	App D	4	
	App D	8	
A	see ABANDON		
ABANDON	4.10.6	Ch.4	33
	Part II	1	
ABANDONED	see ***ABANDONED		
ABEND	Part II	2	
	Part II	70	
	4.10.1	Ch.4	27
abends	4.10.1	Ch.4	27
	see also relevant section of Part II		

access methods 3.3.1 Ch.3 10.1
ALGOL
Delft see ALGOL 60
F see ALGOL 60
W Part II 3
60 Part II 6
68 Part II 9
App E 3
ALL see dynamic allocation
allocation, dynamic 4. Ch.4 1
alphanat 3.1.1 Ch.3 2
ampersands see IF
AND 3.1.4 Ch.3 4
archiving see variable s
arithmetic variables App A
ASCII 3.4.2 Ch.3 12
consoles App D
Assembler Part II 13
AT App F 6
ATTN App D 7

B (blocksize) 4.3.3 Ch.4 10
bad characters App D 5
BADFILE Part II 17
Part II 70
BADP Part II 18
Part II 70
BADPARM Part II 19
Part II 70
BADSYNT Part II 20
Part II 70
basic commands
Phoenix 4.1 Ch.4 2
examples of 4.4 Ch.4 12
BATCH see FAST
BCD 3.2.9 Ch.3 9
BCPL Part II 21
BDAM 3.3.1 Ch.3 10.1
BIGDISC 2.5 Ch.2 13
3.1.6 Ch.3 5

binary mode, paper tape see paper tape
BLKSIZE see blocksizes and DCB parameters
BLOCK 4.3.3 Ch.4 10
blocking factors, formula 3.3.2 Ch.3 11
blocksizes 3.3.2 Ch.3 11
4.3.3 Ch.4 10
BLP 3.2.6 Ch.3 7
bound, credit see credit bound
BPAM 3.3.1 Ch.3 10.1
bpi 3.2.1 Ch.3 5
break 2.2.5 Ch.2 7
App D 2
App D 6
App D 10
4.11.1 Ch.4 35
4.11.1 Ch.4 36
breaking into commands 4.11.1 Ch.4 35
BSAM 3.3.1 Ch.3 10.1

C (cost of a job) App F 2
C command 4.8.1 Ch.4 20
Part II 24
calling level 4.8.3 Ch.4 21
catalogued datasets see permanent datasets
CAMDUMP 4.3.2 Ch.4 7
4.10.1 Ch.4 27
CAMXDUMP 4.3.2 Ch.4 7
4.10.1 Ch.4 27
carriage control characters App H
Cartrifile 3.4.2 Ch.3 13
3.4.4 Ch.3 14
CC see RC
Ce App F 2
CHAIN 2.5 Ch.2 14
2.5 Ch.2 20
App F 6
CHAR 4.11.3 Ch.4 37
Part II 26
character codes App A
see also TRANS
charging 2.7.2 Ch.2 24
CLEAR Part II 27
COBOL Part II 28
codes App A
COLLECT Part II 28.1
COMM 4.3.2 Ch.4 7
command-environment 4.2 Ch.4 3
command-parameter 4.1.2 Ch.4 3
4.2.3 Ch.4 4
command decoder rules App E 4
App E 7
commands
C see C command
COMMAND see C command
conditional see conditional commands
control see control commands
interrupting see breaking
levels of see levels of command
macro see macro commands
nesting see nested commands
program execution see program execution commands
command-sequences 4.8 Ch.4 20
4.8.3 Ch.4 21
4.9.4 Ch.4 25
4.10 Ch.4 27
command-source 4.1.2 Ch.4 2
COMMAND0 4.3.2 Ch.4 7
COMMENT Part II 28.3
Part II 168
comments 4.10.3 Ch.4 29
COMMOUT0 4.3.2 Ch.4 7
2.3.2 Ch.2 10
COMP 2.5 Ch.2 14
COMPARE Part II 28.4
comparisons 4.10.5 Ch.4 32

COMPRESS Part II 29
concatenations
datasets 4.5.9 Ch.4 15
lines see : and CHAR
COND 2.5 Ch.2 14
CONDENSE Part II 31
conditional commands 4.10.3 Ch.4 29
4.10.5 Ch.4 31
see also IF
conditional clauses 4.10.5 Ch.4 31
condition codes see RC
connection, line 2.2.3 Ch.2 4
consoles 2.2.1 Ch.2 3
App D
console sessions 2.2.2ff Ch.2 4
4.11.4 Ch.4 37
App D
error conditions in App D 10
CONTIG 4.3.3 Ch.4 11
control commands, Phoenix 4.1.1 Ch.4 2
4.10.3 Ch.4 29
4.10.6 Ch.4 33
control features 4.10 Ch.4 27
COPIES 2.5 Ch.2 15
COPY Part II 34
cost of job App F 2
see also charging
CREATION Part II 36.1
credit 3.1.3 Ch.3 3
credit bound 3.1.3 Ch.3 3
negative 3.1.3 Ch.3 3
CROSSREF Part II 37
CURRENT Part II 40
current document see %C
current level 4.8.3 Ch.4 21
CYL 3.3.4 Ch.3 12
4.3.3 Ch.4 9
cylinder 3.1 Ch.3 1
DA 3.3.1 Ch.3 10.1
4.3.3 Ch.4 11
data definitions
in Phoenix commands 4.3 Ch.4 5
semantics of 4.5 Ch.4 12
see also ddname
data triples App E 5
dataset name
disc 3.1.1 Ch.3 2
3.1.2 Ch.3 2
magnetic tape 3.2.6 Ch.3 8
datasets
catalogued see permanent datasets
characteristics 4.6 Ch.4 17
instream see here datasets
large see BIGDISC
multi-volume tape see magnetic tape
partitioned see partitioned datasets
permanent see permanent datasets

security	3.1.4	Ch.3	3
temporary	see temporary datasets		
workspace	see %A		
DCB parameters			
simplified	3.3.3	Ch.3	11
tape	3.2.6	Ch.3	7
ddname			
datasets	4.5.5	Ch.4	13
definition of	4.3	Ch.4	5
lists	4.5.10	Ch.4	16
special	see special ddnames		
ddseq	4.3.1	Ch.4	5
DEC GT40	App I		
DECKS	2.5	Ch.2	15
default settings (Phoenix)	4.2.4	Ch.4	5
	see also initial state		
default project	Part II	140.1	
DEFER	3.2.6	Ch.3	8
DEL	see /*DEL		
DELETE	Part II	41	
deleting jobs	see KILL		
DEN	3.2.6	Ch.3	7
density			
disc	3.1	Ch.3	1
magnetic tape	3.2.6	Ch.3	7
device options	App D	3	
dial-up Phoenix characteristics	App D	4	
dial-up number	see telephone number		
dial-up Phoenix	App D	9	
direct access	see DA		
directive triples	App E	5	
DISC	2.5	Ch.2	15
disc drives	3.1	Ch.3	1
disc income	3.1.3	Ch.3	3
DISCIO	2.5	Ch.2	15
disc limit	3.1.3	Ch.3	3
disc space	3.1.3	Ch.3	3
disc track	3.1	Ch.3	1
discs, private	see private discs		
disc units	3.1	Ch.3	1
DISK	see DISC		
DISPLAY	Part II	43	
disposition qualifiers	4.3.3	Ch.4	10
	4.6.3	Ch.4	17
DO	see IF		
dsd	4.3.1	Ch.4	6
concatenations of	4.5.9	Ch.4	15
dsdlist	4.3.1	Ch.4	6
DSN			
disc	3.1.1	Ch.3	2
	3.1.2	Ch.3	2
tape	3.2.6	Ch.3	8
dsname	4.3.1	Ch.4	5
special	4.3.2	Ch.4	7
DUMMY	see %DUMMY		
DUMP	Part II	168	
dynamic allocation	3.	Ch.3	1

EBCDIC	3.4.2	Ch.3	13
	App A		
EDIT	Part II	45ff	
commands	Part II	48	
concatenating lines	Part II	57	
context searches	Part II	57	
control of datasets	Part II	59	
errors	Part II	62	
examples	Part II	63	
fixed format fields	Part II	57	
global exchanges	Part II	57	
justification	Part II	58	
margins	Part II	58	
modes	Part II	60	
operations on current line	Part II	56	
reserved ddnames	Part II	60	
restrictions	Part II	62	
splitting lines	Part II	57	
store requirements	Part II	63	
verification	Part II	61	
EDSORT	Part II	66	
END	Part II	26	
environment-elements	4.2	Ch.4	3
	4.5.10	Ch.4	16
ERRCODE	Part II	168	
ERRJUMP	4.10.7	Ch.4	33
	Part II	69	
error flags	4.10.2	Ch.4	28
error labels	4.10.7	Ch.4	33
errors, flow of control after	4.10.7	Ch.4	33
ERROR	4.10.7	Ch.4	33
	Part II	70	
	see also ***ERROR		
escape character	2.2.5	Ch.2	6
	see also @		
escape combinations	App B		
	App D	3	
EX	see EXAMINE		
EXAMINE	Part II	71	
EXPDT	3.2.6	Ch.3	7
F (format qualifier)	4.3.3	Ch.4	10
	Part II	82	
FA	3.3.3	Ch.3	11
	4.3.3	Ch.4	10
failures, system	2.2.8	Ch.2	8
FALSE	Part II	74	
FAST	Part II	75	
	App F	4	
FB	3.3.2	Ch.3	11
	3.3.3	Ch.3	11
	4.3.3	Ch.4	10
FBA	3.3.3	Ch.3	11
	4.3.3	Ch.4	10
FBS	3.3.1	Ch.3	10.1
	3.3.3	Ch.3	11
	4.3.3	Ch.4	10
FD	4.3.3	Ch.4	10

	Part II	76	
FF	3.3.3	Ch.3	11
	4.3.3	Ch.4	10
FILE	Part II	77	
file protection	see PROTECT		
fileowner	3.1.2	Ch.3	2
FILEPDS	Part II	80.1	
files	see datasets		
file space	see filing system		
filing system	3.1.2	Ch.3	2
	3.1.3	Ch.3	3
FIND	see INFO		
FINISH	2.2.7	Ch.2	8
	4.10.3	Ch.4	29
	4.10.6	Ch.4	33
	Part II	81	
FIXED	4.3.3	Ch.4	10
FIXEDS	4.3.3	Ch.4	10
flags	4.10.2	Ch.4	28
Flexowriter code	App A		
	App B		
FN	4.3.3	Ch.4	10
FORM	Part II	82	
FORTRAN	Part II	83	
format characteristics of a dataset	4.6.2	Ch.4	17
qualifiers	4.3.3	Ch.4	10
FP	3.3.3	Ch.3	11
	4.3.3	Ch.4	10
FTG1	Part II	83	
full-duplex	2.2.1	Ch.2	3
G	see GIGANTIC		
GIGA	see GIGANTIC		
GIGANTIC	4.3.3	Ch.4	9
GLOBAL	see PARS		
global parameters	4.9.1	Ch.4	22
	4.9.4	Ch.4	25
graphicsconsoles	2.2.1	Ch.2	3
	App I		
GT 40	App I		
G1	see FORTRAN		
half-duplex	2.2.1	Ch.2	3
HASP	1.	Ch.1	1
system log	2.6.1	Ch.2	21
terminators	2.3.4	Ch.2	11
HELP	Part II	90	
HELPK	Part II	90.1	
here datasets	4.5.8	Ch.4	15
hexadecimal	App A		
H-extended	Part II	83	
HOLDSP	Part II	168	
HP	Part II	91	
IEBCOPY	see IBMCOPY		
IEHMOVE	see TRANSFER		
IBMCMPRS	Part II	91.1	
IBMCOPY	Part II	92	

IBM utilities App G
IF 4.10.5 Ch.4 31
Part II 94
IMAGE Part II 95
income, disc see disc income
INFO Part II 99
inhibited fileowner 3.1.3 Ch.3 3
INIT see MKINIT
initial state 4.11.4 Ch.4 37
inky pens (plotter) 3.5.1 Ch.3 17
INPUT Part II 101
instream datasets see also here datasets
in command sequences 4.8.3 Ch.4 21
int 4. Ch.4 1
App E 3
inter-block gap
disc 3.3.2 Ch.3 10.1
tape 3.2.9 Ch.3 9
3.3.2 Ch.3 10.1
inter-record gap see inter-block gap
internal readers 2.1 Ch.2 2
2.3.3 Ch.2 10
see also SUBMIT
INTRDR 2.3.3 Ch.2 10
ISAM 3.3.1 Ch.3 10.1

JCL (JD terminator) 2.5 Ch.2 16
JCL jobs 2.3.1 Ch.2 9
2.3.3 Ch.2 10
output from 2.6.1 Ch.2 21
JD see job description
job description 2.4 Ch.2 12
statements 2.5 Ch.2 13
JOBLIB 4.3.2 Ch.4 7
job limits 2.8.1 Ch.2 24
job queue position App F 2
job sources 2.1 Ch.2 2
job title 2.4.1 Ch.2 13
job types 2.3.1 Ch.2 9
jobs charging formula 2.7.2 Ch.2 24
cost App F 2
end App F 6
estimated cost App F 2
execution App F 6
JCL see JCL jobs
Phoenix see Phoenix jobs
saved see SAVE
submission App F 6
JOB statement 2.4.1 Ch.2 12
JUMP 4.10.4 Ch.4 30
Part II 102
jump commands 4.10.3 Ch.4 29
jumps 4.10.4 Ch.4 30

K 4.3.3 Ch.4 9
K (scheduling App F 5
K (space) 4.3.3 Ch.4 9
see also SPACE

KET Part II 26
keyword 4.7 Ch.4 19
keyword-value-pair 4.7 Ch.4 19
KILL Part II 102.1
KILLJOB see KILL

L (LRECL) 4.3.3 Ch.4 10
languages
ALGOL W Part II 3
ALGOL, Delft see ALGOL 60
ALGOL F see ALGOL 60
ALGOL 60 Part II 6
ALGOL 68 Part II 9
Assembler Part II 13
BCPL Part II 21
COBOL Part II 28
FORTRAN Part II 83
MLI Part II 121
PASCAL Part II 129
PLI Part II 132
SNOBOL Part II 155
SPITBOL see SNOBOL
LA see LARGE
LABEL 3.2.6 Ch.3 7
Part II 102.2
labels 4.10.4 Ch.4 30
LARGE 4.3.3 Ch.4 9
level see %LEVEL
levels of command 4.8.3 Ch.4 21
libraries, macro 4.11.5 Ch.4 38
App E
LIBRARY 4.11.5 Ch.4 36
Part II 103
limit, disc see disc limit
LIMSTORE 2.5 Ch.2 16
line concatenation Part II 26
line connection 2.2.3 Ch.2 4
line speeds 2.2.1 Ch.2 3
App D 9
line width, consoles 2.2.9 Ch.2 8
linkage editor Part II 104
LIST Part II 106
LISTPDS see PRINT
LKED see linkage editor
LOCAL Part II 112
local parameters 4.9.1 Ch.4 22
4.9.4 Ch.4 25
LOGOFF see FINISH
LOGON 2.2.4 Ch.2 5
Part II 113
LRECL 4.3.3 Ch.4 10

M App F 3
App F 5
macro-commands 4.7 Ch.4 18
App E
macro-definition 4.7 Ch.4 19
macro libraries 4.11.5 Ch.4 38

	App E	7	
macro-name	4.7	Ch.4	19
magnetic tape	3.2ff	Ch.3	5
access states	3.2.10	Ch.3	10
application for	3.2.2	Ch.3	6
DCB parameters	3.2.6	Ch.3	7
dataset names	3.2.6	Ch.3	8
densities	3.2.1	Ch.3	5
	3.2.6	Ch.3	7
	3.2.9	Ch.3	9
drives	3.2.1	Ch.3	5
inter-block gap	see inter-block gap		
jobs	3.2.5	Ch.3	6
labels	3.2.4	Ch.3	6
	3.2.6	Ch.3	7
marks	3.2.7	Ch.3	8
multi-volume	3.2.6	Ch.3	8
names	3.2.4	Ch.3	6
	3.2.6	Ch.3	8
parity	3.2.1	Ch.3	5
	3.2.6	Ch.3	7
	3.2.9	Ch.3	9
registration	3.2.2	Ch.3	6
rings	3.2.9	Ch.3	10
scratch	3.2.8	Ch.3	9
standard labelled	3.2.4	Ch.3	6
standard specification	3.2.9	Ch.3	9
tape marks	3.2.6	Ch.3	8
transferring	3.2.9	Ch.3	9
units (9-track)	3.2.1	Ch.3	5
(7-track)	3.2.1	Ch.3	5
unlabelled	3.2.4	Ch.3	6
write-permit rings	3.2.10	Ch.3	10
MC	see PRINT		
MED	see MEDIUM		
MEDIUM	4.3.3	Ch.4	9
MERGE	see SORT		
merit value	App F 3		
MESSAGE	Part II	117	
messages document	see %M		
miscellaneous qualifiers	4.3.3	Ch.4	11
MKINIT	Part II	119.1	
MKLIB	Part II	120	
MLI	Part II	121	
MOD	4.3.3	Ch.4	10
monitor	see Phoenix Monitor		
MSGCLASS	2.5	Ch.2	16
MSGLEVEL	2.5	Ch.2	16
mylar film	3.5.1	Ch.3	17
N (qualifier)	4.3.3	Ch.4	10
N (scheduling)	App F 3		
NA	App E 5		
name	4.	Ch.4	1
	App E 3		
negative credit	3.1.3	Ch.3	3
nested commands	4.8.3	Ch.4	21
NEW	4.3.3	Ch.4	10

NEWPDS
NIOP 3.3.1 Ch.3 10.1
NL (qualifier) see NORMAL
NO Part II 123
NOCONTIG 4.3.3 Ch.4 11
NODA 4.3.3 Ch.4 11
NODEF App E 5
NOEXEC Part II 168
NOPMON 2.5 Ch.2 16
NOPWREAD 3.2.6 Ch.3 7
NORLSE 4.3.3 Ch.4 11
NORMAL 4.3.3 Ch.4 9
NOSTOP Part II 168
NOSUB 4.9.3 Ch.4 24
Part II 168
NOSUBH Part II 168
NOTIFY 2.5 Ch.2 16
Part II 123.1
NOTOPT 4.3.3 Ch.4 11
App E 6
NOW App F 3
NULL 4.10.6 Ch.4 33
Part II 124
NULLEKET App E 5

OLD 4.3.3 Ch.4 10
OLDABEND Part II 125
ONLINE Part II 126
ONLINE SYSTEM FULL 2.2.4 Ch.2 5
operating system 2.1 Ch.2 1
OPT 4.3.3 Ch.4 11
App E 5
OPTNA App E 5
OPTNAND App E 5
OR see IF
organisational qualifiers 4.3.3 Ch.4 11
4.6.4 Ch.4 18
OS 1. Ch.1 1
output document see %O
output, system 2.6 Ch.2 21
outstanding work App F 2
OVERNIGHT App F 3

P see PARS
see PRIORITY
PAGE (JD parameter) 2.5 Ch.2 17
paper tape 3.4ff Ch.3 12.1
input 3.4.1 Ch.3 12.1
binary 3.4.4 Ch.3 14
3.4.5 Ch.3 15
codes 3.4.2 Ch.3 12.1
console 2.2.9 Ch.2 9
errors 3.4.3 Ch.3 13
normal mode 3.4.5 Ch.3 14
options 3.4.5 Ch.3 14
stream mode 3.4.4 Ch.3 14
3.4.5 Ch.3 15
translated

binary mode	3.4.5	Ch.3	15
warning sequences	3.4.4	Ch.3	13
jobs	3.4.6	Ch.3	16
	3.4.9	Ch.3	16
output	3.4.7	Ch.3	16
binary	3.4.9	Ch.3	16
codes	3.4.8	Ch.3	16
dd statements	3.4.7	Ch.3	16
	3.4.9	Ch.3	16
JCL	3.4.9	Ch.3	16
warning sequences	3.4.4	Ch.3	13
widths	3.4.2	Ch.3	12.1
	3.4.8	Ch.3	16
parameter			
data	4.2.3	Ch.4	4
setting	4.9.2	Ch.4	22
substitution	4.9.3	Ch.4	24
values	4.9.2	Ch.4	22
	4.9.4	Ch.4	25
parameter-declaration	4.8.1	Ch.4	20
	4.9.2	Ch.4	22
error in	Part II	19	
parameter-name	4.9.1	Ch.4	22
	Part II	127	
parameter-occurrences			
	4.8.3	Ch.4	21
	4.9.3	Ch.4	24
errors in	Part II	18	
in here datasets	4.5.8	Ch.4	15
parity	see magnetic tape		
PARS	4.8.1	Ch.4	20
	4.8.2	Ch.4	20
	Part II	127	
partitioned datasets	3.3.1	Ch.3	10.1
	4.5.7	Ch.4	14
PASCAL	Part II	129	
PASSWORD			
JD	2.5	Ch.2	17
tape	3.2.6	Ch.3	7
(command)	Part II	129.1	
PATCH	Part II	130	
pds	see partitioned datasets		
PDS	see PO		
PDSRECOV	Part II	131.1	
PDSSCAN	Part II	131.3	
permanent datasets	3.1.2	Ch.3	2
	3.1.3	Ch.3	2
	4.5.2	Ch.4	12
PHOENIX (JD terminator)	2.5	Ch.2	18
Phoenix			
command language	4.	Ch.4	1ff
console sessions	see console sessions		
jobs	2.3.1	Ch.2	9
	2.3.2	Ch.2	10
output from	2.6.1	Ch.2	21
interface	App D		

Monitor 2.3.1 Ch.2 10
limits 2.8.1 Ch.2 25
output from 2.6.2 Ch.2 22
qualifiers,disposition see disposition qualifiers
format see format qualifiers
organisation see organisation qualifiers
size see size qualifiers
PHOENIX NOT AVAILABLE 2.2.4 Ch.2 5
PHOENIX REENTERED 4.11.1 Ch.4 36
PHOENIX STARTING App D 10
PHX see PHOENIX
PI App F 2
PLI Part II 132
PLOTTER 2.5 Ch.2 18
plotter 2.3.3 Ch.2 11
2.8.2 Ch.2 25
3.5.1 Ch.3 17
PO 4.3.3 Ch.4 11
4.5.7 Ch.4 14
4.6.4 Ch.4 18
POST 2.5 Ch.2 18
previous current document see %P
PRINT Part II 139
PRINTER 2.5 Ch.2 18
printer 2.3.3 Ch.2 11
3.5.2 Ch.3 17
chain App C 1
PRIORITY 2.5 Ch.2 19
priority App F 3
priority limit App F 2
default App F 5
PRIVATE 3.2.6 Ch.3 8
private discs 3.1.5 Ch.3 4
PRIVATE LIBRARY CANCELLED 4.11.1 Ch.4 36
procedures
Phoenix 4.8.2 Ch.4 20
see also languages
JCL 2.3.1 Ch.2 9
2.3.3 Ch.2 10
see also languages
program execution commands 4.1.2 Ch.4 2
PROJECT Part II 140.1
project number 2.4.1 Ch.2 12
2.7 Ch.2 23
PROTECT Part II 140.3
PS 4.3.3 Ch.4 11
4.6.4 Ch.4 18
pseudo quotes App E 4
PUNCH Part II 141
PUNCH (JD parameter) 2.5 Ch.2 19
PUNCH O PURGE Part II 163
Q see QUIT
qds
definition of 4.3.1 Ch.4 6
example of 4.3.3 Ch.4 12
see also qualifier

qlist, definition of 4.3.1 Ch.4 6
QREST App E 3
QSAM 3.3.1 Ch.3 10.1
QTEXT App E 3
qualifier 4.3.3 Ch.4 9
defaults 4.2.4 Ch.4 5
definition of 4.3.1 Ch.4 6
disposition see disposition qualifier
format see format qualifier
miscellaneous see miscellaneous qualifier
organisational see organisation qualifiers
overriding rules 4.3.3 Ch.4 12
size see size qualifiers
qualifier-terminator,definition of 4.3.1 Ch.4 6
QUIET Part II 168
QUIT 4.10.6 Ch.4 33
Part II 143
QUOTE Part II 26
quoted-text 4. Ch.4 1
see also QTEXT

rate of working App F 2
RC 4.10.2 Ch.4 28
Part II 70
Part II 143.1
READ/WRITE 3.2.10 Ch.3 10
readers 2.1 Ch.2 2
READ ONLY 3.2.10 Ch.3 10.1
READY 2.2.4 Ch.2 6
4.8.3 Ch.4 21
READY. 4.11.1 Ch.4 35
real time limit 2.8.1 Ch.2 24
RECFM see DCB parameters
REFLECT Part II 168
region see store requirement
RELATION Part II 143.11
RELEASE Part II 143.12
RENAME Part II 143.14
REQUEST Part II 143.16
REROUTE Part II 143.18
reserved ddnames see special ddnames
dsnames see special dsnames
resource allocation 2.7 Ch.2 23
control 2.7.1 Ch.2 23
units 2.7.2 Ch.2 24
REST App E 3
RESTORE Part II 168
restricted facilites 2.8.2 Ch.2 25
utilities App G
results document see %R
RETAIN 3.2.6 Ch.3 8
RETPD 3.2.6 Ch.3 7
RETURN 4.10.4 Ch.4 31
Part II 144
return code see RC
RJE stations see routes
RLSE 4.3.3 Ch.4 11
ROUTE 2.5 Ch.2 19

routes 2.5 Ch.2 19
RUBOUT Part II 144.1
RUN Part II 145

S (set user) 4.11.2 Ch.4 36
4.11.4 Ch.4 37
Part II 27

S (SMALL) see SMALL
SAVE Part II 146.1
scheduling App F
SCRATCH Part II 147
scratch tapes see magnetic tapes
SEND Part II 148
sequential datasets see PS
session interruption
SET Part II 151
commands 4.11.2 h.4 36
IF parm IS SET 4.10.5 Ch.4 31
SETUP Part II 153
shares 2.7.1 Ch.2 23
App F
SHR 4.3.3 Ch.4 10
size
characteristics of dataset 4.6.1 Ch.4 17
file see filing system
qualifiers 3.3.4 Ch.3 12
4.3.3 Ch.4 9
SKIP 4.10.4 Ch.4 30
Part II 154
SMALL 4.3.3 Ch.4 9
SNOBOL Part II 155
SORT Part II 157
SPACE 4.3.3 Ch.4 9
Part II 159
see also filing system
special ddnames 4.3.2 Ch.4 7
special dsnames 4.3.2 Ch.4 7
special parameters 4.9.5 Ch.4 25
,system 4.9.5 Ch.4 27
,user 4.9.5 Ch.4 26
SPITBOL see SNOBOL
SPOOL Part II 160
Part II 168
SPSS Part II 161
status qualifiers see disposition qualifiers
STEPLIB 4.3.2 Ch.4 7
store requirement Phoenix commands) 4.2.1 Ch.4 4
see also Part II
STATUS Part II 163
stream mode paper tape 3.4.5 Ch.3 15
SUBMIT Part II 165
SUGGEST Part II 166.1
supressing printing App B
App D 4
SW see SWITCH
SWITCH Part II 167
see also switches

switches 4.10.2 Ch.4 28
Part II 167
syntactic elements 4. Ch.4 1
symbolic parameters see relevant language
4.3.1 Ch.4 5
syntax errors in commands Part II 20
SYSABEND 4.10.1 Ch.4 27
SYSOUT 2.3.3 Ch.2 11
system commands 4.3.2 Ch.4 7
completion codes see abends and %ABEND
documents 4.5.6 Ch.4 13
failures (effect at consoles) 2.2.8 Ch.2 8
output 2.6 Ch.2 21
SYSUDUMP 4.10.1 Ch.4 27

tab App D 2
see also IMAGE
TAPECOPY Part II 169
tape
magnetic see magnetic tape
marks 3.2.7 Ch.3 8
paper see paper tape
TAPESCAN Part II 172.1
TAPESTAT Part II 172.4
TAPE7 2.5 Ch.2 20
TAPE9 2.5 Ch.2 20
telephone number (Phoenix dial-up) 2.2.3 Ch.2 4
teletypes App D 1
temporary dataset names 3.1.1 Ch.3 2
temporary datasets 3.1.1 Ch.3 1
4.5.3 Ch.4 12
terminals see consoles
TERMINAL Part II 173
see also INFO
termination commands 4.10.3 Ch.4 29
4.10.6 Ch.4 33
TERSE Part II 168
TESTTERM Part II 174
text 4. Ch.4 1
App E 3
TIME see COMP
time of day see INFO and %
time requirement (Phoenix commands) 4.2.2 Ch.4 4
Titan Flexowriter code App A
TLS Part II 176
TLSCOPY Part II 183
TLSINIT Part II 186
TLSRECOV Part II 187
TLSSCAN Part II 189.1
TO Part II 189.3
TR see TURNROUND
track 3.3.2 Ch.3 11
TRANS Part II 190
TRANSFER Part II 193.1
3.3.4 Ch.3 12
TRK 4.3.3 Ch.4 9
TRUE Part II 194
4.10.2 Ch.4 28

TRTCH 3.2.6 Ch.3 7
TURNROUND 2.5 Ch.2 20
App F 3
TYPE Part II 195

U (DCB parameter) 3.3.2 Ch.3 11
4.3.3 Ch.4 10
U (usage) App F 2
U (userid) 4.11.4 Ch.4 37
Part II 27
see also userid
UNIT 2.3.3 Ch.2 10
3.2.6 Ch.3 7
UNLESS 4.10.5 Ch.4 31
Part II 196
UNSET 4.10.5 Ch.4 31
unsetting parameters Part II 128
unsetting switches Part II 167
UPPER Part II 168
usage App F 2
userid 2.4.1 Ch.2 12
see also U
user identifier see userid
switches 4.10.2 Ch.4 28
Part II 197

utilities
COLLECT Part II 28.1
COMMENT Part II 28.3
COMPARE Part II 28.4
COMPRESS Part II 29
CONDENSE Part II 31
COPY Part II 34
CREATION Part II 36.1
CROSSREF Part II 37
CURRENT Part II 40
DELETE Part II 41
DISPLAY Part II 43
EDIT Part II 45
EDSORT Part II 66
EXAMINE Part II 71
FILE Part II 77
FILEPDS Part II 80.1
GLOBALS see PARS
HELP Part II 90
HELPK Part II 90.1
HP Part II 91
IBMCMPRS Part II 91.1
IBMCOPY Part II 92
IMAGE Part II 95
INFO Part II 99
INPUT Part II 101
KILL Part II 102.1
LABEL Part II 102.2
LIBRARY Part II 103
Linkage Editor Part II 104
LIST Part II 106
LOCAL Part II 112
MESSAGE Part II 117

MKINIT	Part II	119.1	
MKLIB	Part II	120	
NOTIFY	Part II	123.1	
NULL	Part II	124	
PARS	Part II	127	
PASSWORD	Part II	129.1	
PATCH	Part II	130	
PDSRECOV	Part II	131.1	
PDSSCAN	Part II	131.1	
PRINT	Part II	139	
PROJECT	Part II	140.1	
PROTECT	Part II	140.3	
PUNCH	Part II	141	
RELATION	Part II	143.11	
RELEASE	Part II	143.12	
RENAME	Part II	143.14	
REQUEST	Part II	143.16	
REROUTE	Part II	143.18	
restricted	App G		
RUBOUT	Part II	144.1	
RUN	Part II	145	
SAVE	Part II	146.1	
SCRATCH	Part II	147	
SEND	Part II	148	
SET	Part II	151	
SETUP	Part II	153	
SORT	Part II	157	
SPOOL	Part II	160	
SPSS	Part II	161	
STATUS	Part II	163	
SUBMIT	Part II	165	
SUGGEST	Part II	166.1	
TAPECOPY	Part II	169	
TAPESCAN	Part II	172.1	
TAPESTAT	Part II	172.4	
TESTTERM	Part II	174	
TLS	Part II	183	
TLSINIT	Part II	186	
TLSRECOV	Part II	187	
TLSSCAN	Part II	189.1	
TO	Part II	189.3	
TRANS	Part II	190	
TRANSFER	Part II	193.1	
TYPE	Part II	195	
VIEW	Part II	198.1	
XLABEL	Part II	199.1	
XMODULE	Part II	199.3	
XPDS	Part II	200	
XTAPE	Part II	207	
VAR	4.3.3	Ch.4	10
VARIABLE	4.3.3	Ch.4	10
variables	Part II	198	
VB	3.3.2	Ch.3	11
	3.3.3	Ch.3	12
	4.3.3	Ch.4	10
VBA	3.3.3	Ch.3	12
	4.3.3	Ch.4	10

VBL	3.3.3	Ch.3	12
	4.3.3	Ch.4	10
VBS	3.3.2	Ch.3	11
	3.3.3	Ch.3	12
	4.3.3	Ch.4	10
VBSL	3.3.3	Ch.3	12
	4.3.3	Ch.4	10
VBST	3.3.3	Ch.3	12
	4.3.3	Ch.4	10
VBX	3.3.3	Ch.3	12
	4.3.3	Ch.4	10
VIEW	Part II	198.1	
VOL	3.2.6	Ch.3	8
	4.3.3	Ch.4	11
w	App F 2		
	App F 5		
WAIT	Part II	199	
wait limit	2.8.1	Ch.2	24
	2.8.1	Ch.2	25
WATFIV	Part II	83	
wide plotter	see plotter		
word	4.	Ch.4	1
	App E 3		
work, outstanding	App F 2		
rate of	App F 2		
workspace datasets	see %A		
workstations	see routes		
write-permit ring	3.2.6	Ch.3	7
	3.2.10	Ch.3	10
WRITE ON REQUEST	3.2.10	Ch.3	10
XLABEL	Part II	199.1	
XMODULE	Part II	199.3	
XPDS	Part II	200	
XTAPE	Part II	207	
YES	Part II	209	
2741 terminals	App D 7		